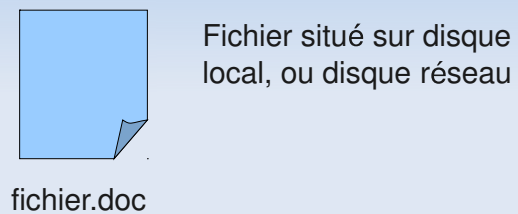


Contrôle de versions et travail collaboratif

Problématique

Organisation du travail collaboratif

- Problème de l'accès concurrent à un fichier



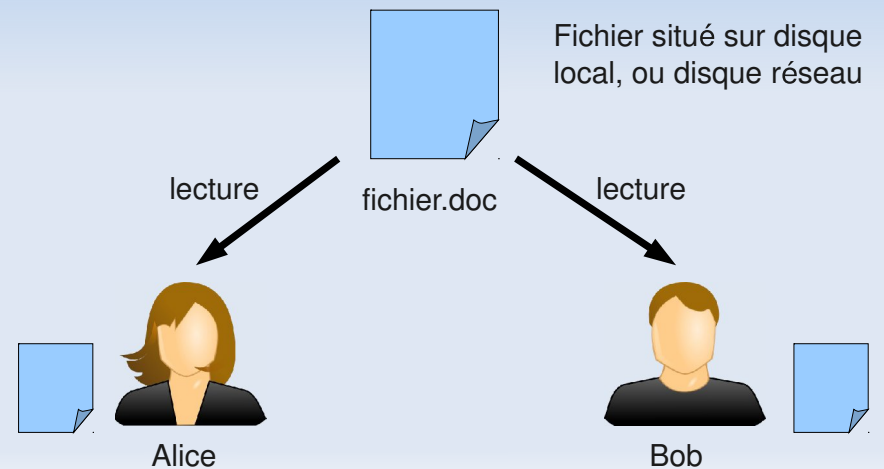
Alice



Bob

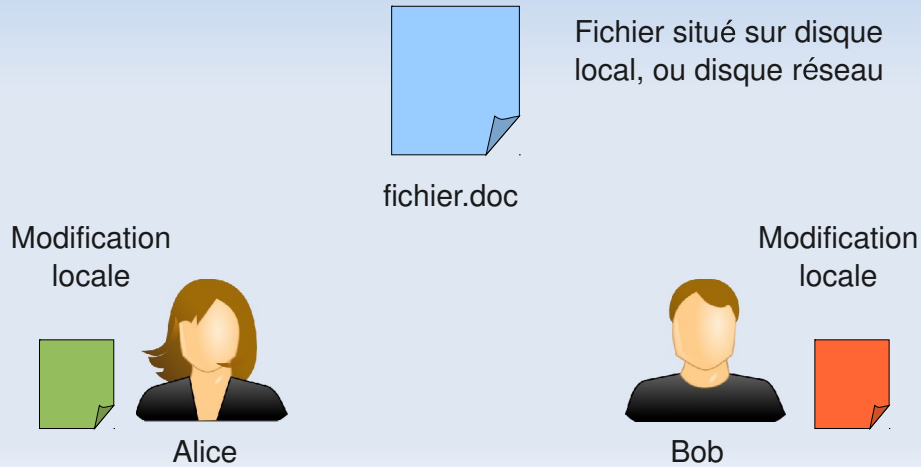
Organisation du travail collaboratif

- Problème de l'accès concurrent à un fichier



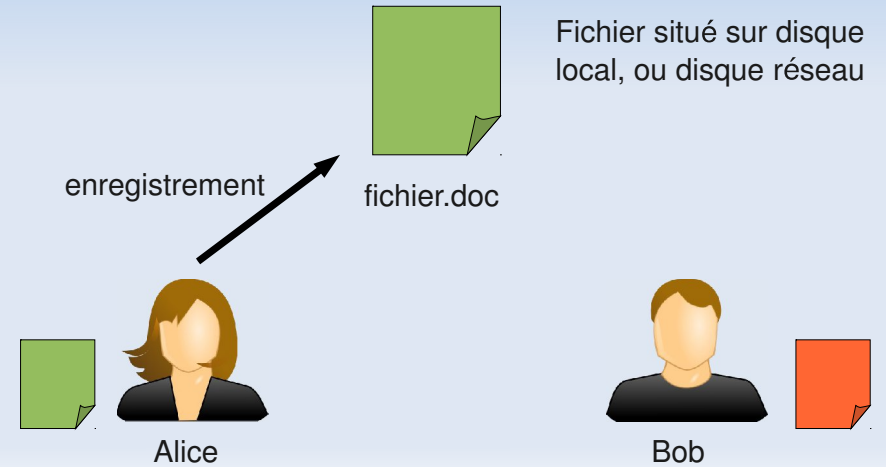
Organisation du travail collaboratif

- Problème de l'accès concurrent à un fichier



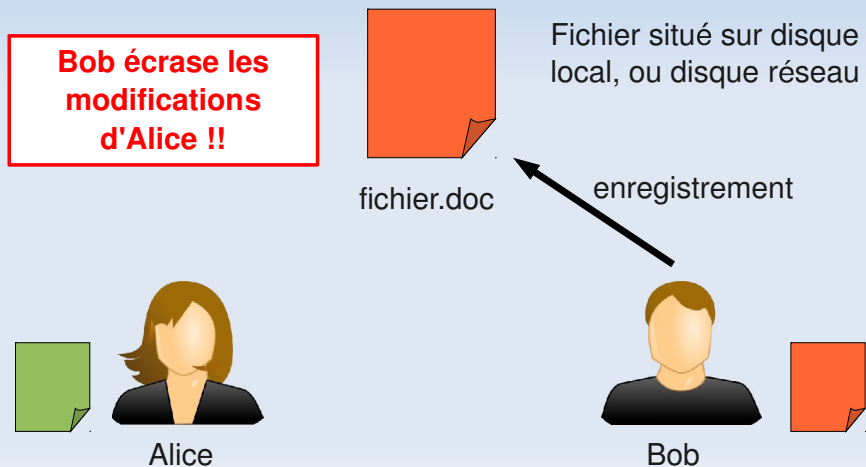
Organisation du travail collaboratif

- Problème de l'accès concurrent à un fichier



Organisation du travail collaboratif

- Problème de l'accès concurrent à un fichier



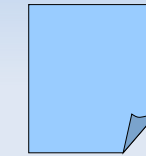
Organisation du travail collaboratif

Organisation du travail collaboratif

- Édition simultanée d'un document
la solution «verrouillage, édition, déverrouillage»
 - instauration d'un mécanisme de verrou sur fichier
 - un utilisateur souhaite modifier un fichier :
 - mise en place d'un verrou avant édition
 - déverrouillage après enregistrement des modifications sur le fichier
 - si verrou présent : accès impossible
 - un seul utilisateur modifie un fichier à la fois
 - garantit l'intégrité des modifications

Organisation du travail collaboratif

- Verrouillage, Édition, Déverrouillage



fichier.doc

Fichier situé sur disque local, ou disque réseau



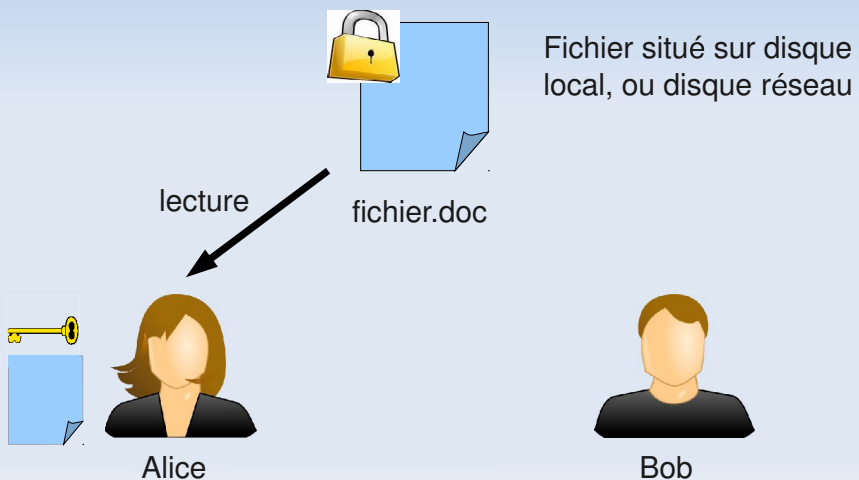
Alice



Bob

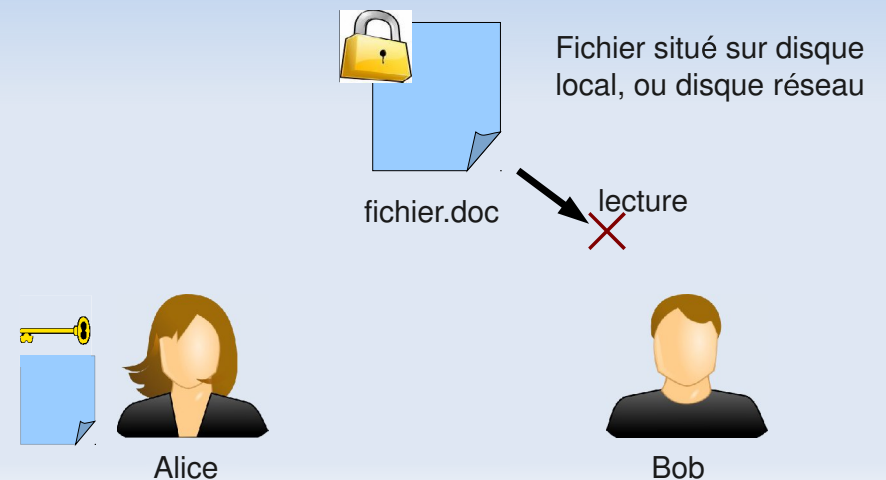
Organisation du travail collaboratif

- Verrouillage, Édition, Déverrouillage



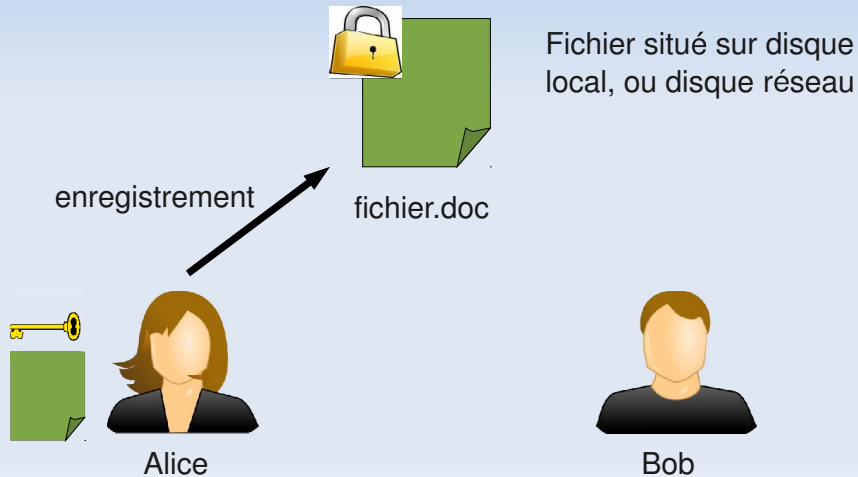
Organisation du travail collaboratif

- Verrouillage, Édition, Déverrouillage



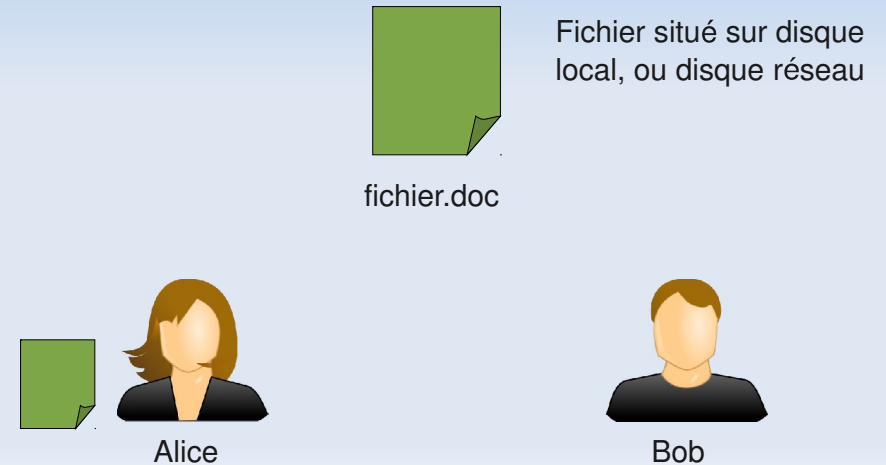
Organisation du travail collaboratif

- Verrouillage, Édition, Déverrouillage



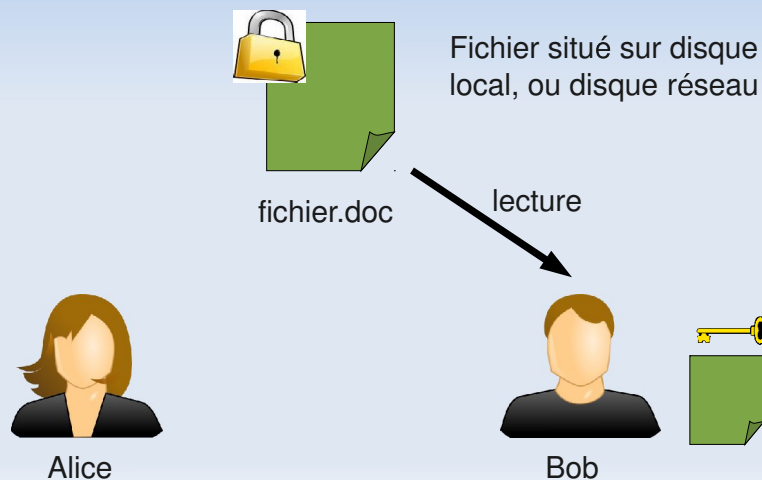
Organisation du travail collaboratif

- Verrouillage, Édition, Déverrouillage



Organisation du travail collaboratif

- Verrouillage, Édition, Déverrouillage



Organisation du travail collaboratif

- Édition simultanée d'un document
la solution «verrouillage, édition, déverrouillage»
 - garantit qu'une modification ne sera pas écrasée
 - ... mais nombreuses restrictions :
 - Problème du verrou oublié :
nécessité de l'intervention de l'administrateur
 - Faux sentiment de sécurité :
verrou sur un fichier seulement
dépendances entre les fichiers ignorées
 - Mise en place de verrous inutiles

Organisation du travail collaboratif

- Édition simultanée d'un document
la solution «copie, modification, fusion»
 - des utilisateurs multiples travaillent sur des copies du fichier, soumettent leur modifications
 - lors de l'envoi de la nouvelle version du fichier :
 - le fichier n'a pas été édité entretemps : MAJ ok
 - le fichier a été édité :
 - fusion des fichiers
 - notification de conflit, intervention de l'utilisateur requise pour les corriger

Organisation du travail collaboratif

- Historique :
 - **SCCS : Source Code Control System**
 - 1972, Marc J. Rochkind, Laboratoire Bell
 - Gestion de plusieurs versions d'un fichier
 - **CVS : Concurrent Versions Subsystem**
 - 1986, Dick Grune
 - Logiciel Open Source
 - Permet la gestion *concurrente* de versions
 - Architecture centralisée
 - Utilisé pendant des années
 - Beaucoup de failles

Organisation du travail collaboratif

- Quelle solution de travail collaboratif ?
 - **Verrouillage, édition, déverrouillage :**
 - si deux copies de travail ne peuvent se fusionner. Ex: fichiers binaires : images, vidéos, ...
 - **Copie, modification, fusion :**
 - idéal pour réelle collaboration
 - Basé sur l'idée que des travaux concurrents peuvent être fusionnés.
 - Souvent le cas sur fichiers texte et fichiers ASCII
Ex : code source de programme
 - Idéal : système combinant les deux solutions

Organisation du travail collaboratif

- Historique :
 - **SVN : Subversion**
 - 2000, CollabNet
 - Successeur de CVS
 - conçu sur le design de CVS : le modèle est bon, mais l'implémentation est en cause.
 - corrige les failles de CVS
 - devenu une référence

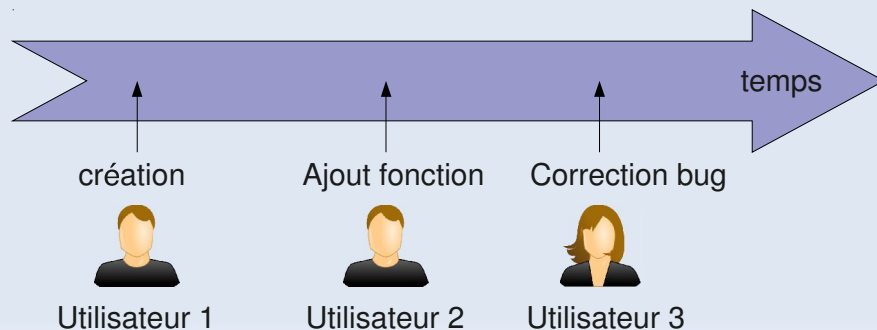
Présentation de Subversion

Présentation de Subversion

- Système de gestion de versions de fichiers
- Permet :
 - la récupération d'une version antérieure d'un fichier
 - d'examiner l'historique des changements
 - déterminer quand un document a été modifié
 - de trouver qui est à l'origine d'une modification
 - ...

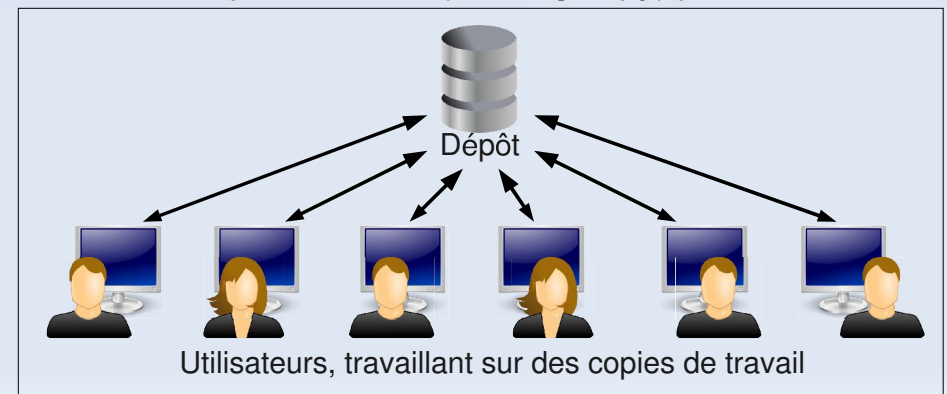
Présentation de Subversion

- Analogie : machine à voyager dans le temps
 - Pour un fichier donné (exemple code source .c) :



Présentation de Subversion

- Architecture centralisée type «client-serveur »
 - Un dépôt (repository) : stockage du projet
 - Une copie de travail (working copy) par utilisateur



Présentation de Subversion

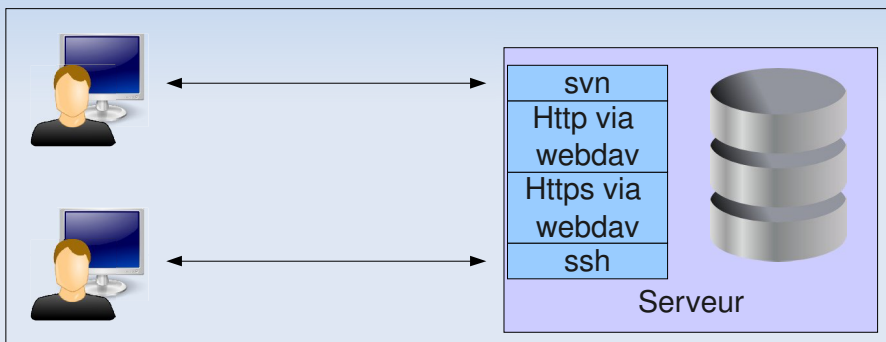
- Chaque sous-répertoire du projet contient un répertoire spécial **.svn** :
- Répertoire d'administration
- Maintenu par Subversion
- Aide Subversion à :
 - reconnaître les fichiers avec modifications locales
 - déterminer les fichiers obsolètes / dépôt
- Répertoire caché. Affichage : **ls -a**

Présentation de Subversion

- Identification d'un dépôt :
 - Modèle URL : Universal Ressource Locator
- Accès à un dépôt distant :
 - (Protocole://) (serveur) [:port] (emplacement dépôt)

Présentation de Subversion

- Accès à un dépôt :



- Cas particulier : dépôt et copie de travail sur la même machine → accès par protocole «file://»

Commandes de base de Subversion

Commandes de base de Subversion

- Utilisation de Subversion coté utilisateur
 - Principale commande : **svn**
 - Utilise des sous-commandes
 - Création d'une copie de travail
 - Soumettre des modifications
 - Récupérer les modifications des autres utilisateurs
 - Résoudre l'éventuelle apparition de conflits
 - Verrouiller / déverrouiller un fichier (exclusivité)
 - ...

Commandes de base de Subversion

- Obtenir de l'aide :
svn help

```
$ svn help
```

```
usage : svn <sous-commande> [options] [param]  
Client texte interactif de Subversion
```

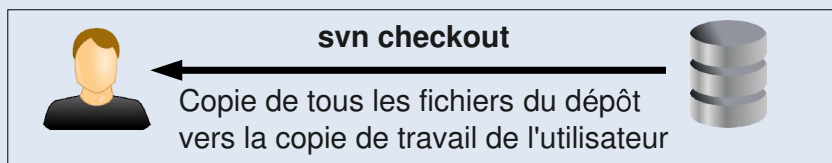
```
Entrer svn help <sous-commande> pour l'aide sur  
une sous commande
```

```
Sous-commandes disponibles :
```

add	checkout (co)	cleanup
commit (ci)	copy (cp)	delete
diff (di)	export	lock
mkdir	move	resolve ...

Commandes de base de Subversion

- Créer une copie de travail depuis un dépôt :
svn checkout
 - Paramètre : l'url du dépôt
 - Crée un répertoire sur le poste de l'utilisateur
 - Y copie l'ensemble des fichiers du dépôt



Commandes de base de Subversion

- Exemple :

```
$ svn checkout svn://monServ.org/var/svn/jeuEchec
```

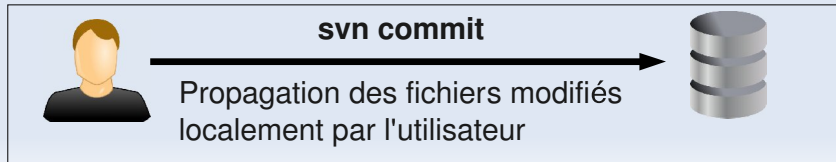
```
A jeuEchec/  
A jeuEchec/plateau.cpp  
A jeuEchec/Piece.cpp  
A jeuEchec/plug-in/  
A jeuEchec/plug-in/IA.cpp  
A jeuEchec/jeuEchec/GUI.cpp
```

```
Révision 1 extraite
```

- 'A' : éléments ajoutés à la copie de travail

Commandes de base de Subversion

- Soumettre des modifications locales au dépôt : **svn commit**
 - «Propager» une modification locale
 - Paramètres :
 - liste des fichiers à propager.
 - Si aucun fichier spécifié, propagation de toute l'arborescence à partir du répertoire courant
 - message décrivant les modifications : **-m**



Commandes de base de Subversion

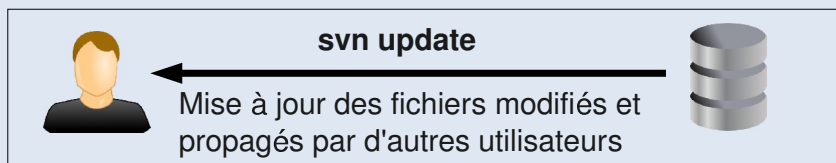
- Exemple :

```
$ svn commit Plateau.cpp -m « Correction de bug »  
Envoi      Plateau.cpp  
Transmission des données .....  
Révision 2 propagée
```

- Envoi de Plateau.cpp au dépôt correspondant
 - Nom du dépôt : enregistré dans **.svn**
 - Si dépôt protégé, demande de mot de passe

Commandes de base de Subversion

- Mettre à jour les fichiers depuis le dépôt : **svn update**
 - Paramètres :
 - liste des fichiers à mettre à jour.
 - Si aucun fichier spécifié, mise à jour de toute l'arborescence à partir du répertoire courant



Commandes de base de Subversion

- Exemple

```
$ svn update  
U   plateau.cpp  
U   plug-in/IA.cpp  
Actualisé à la révision 3
```

- 'U' : fichier mis à jour
- Seuls les fichiers modifiés par d'autres utilisateurs sont importés

Commandes de base de Subversion

- Obtenir l'aide d'une sous-commande :
svn help <sous-commande>

```
$ svn help commit
```

```
commit (ci): Envoie les modification de votre copie de travail vers le dépôt.
```

```
usage : commit [CHEMIN...]
```

```
Un message, éventuellement vide, doit être fourni pour le journal. S'il n'est pas donné par --message ou --file, un éditeur est lancé.
```

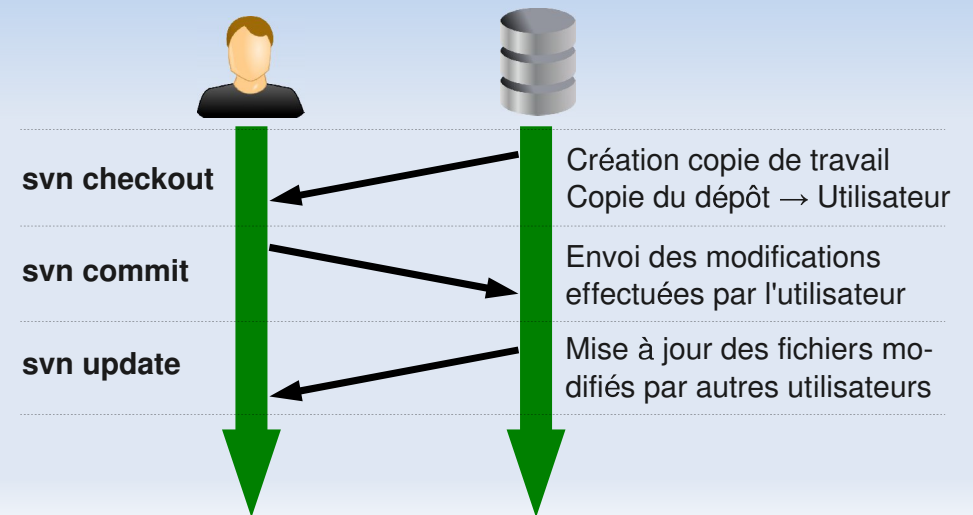
```
Options valides:
```

```
...
```

Gestion des révisions

Commandes de base de Subversion

- En résumé :



Gestion des révisions

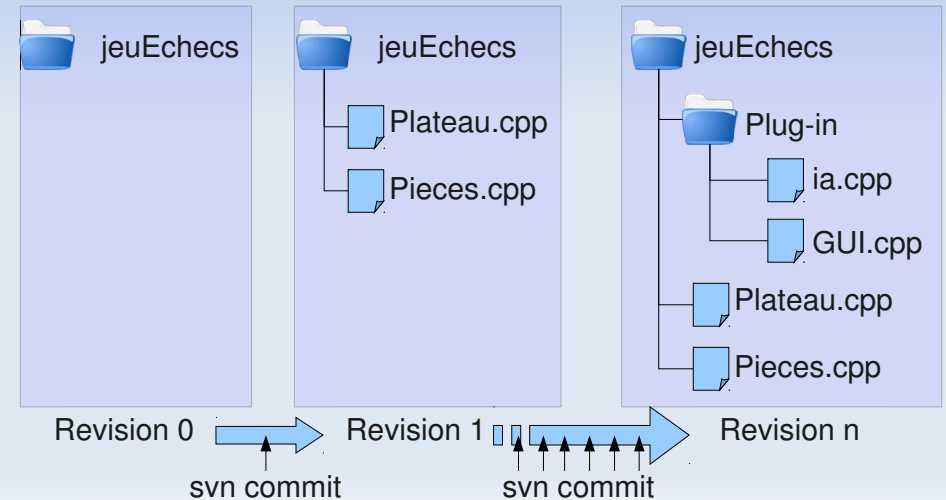
- svn commit** propage les changements sur un ou plusieurs fichiers / répertoires
- Opération atomique :
 - vu comme une seule « évolution » du projet
 - soit tous les changements sont opérés en même temps, soit aucun ne l'est
- Chaque fois que le dépôt accepte la propagation d'une modification :
 - nouvel état du projet → révision

Gestion des révisions

- **Une révision**
 - Correspond à l'état d'un projet à un moment donné
 - Numérotée
 - Incrémental en partant de 0
 - Révision 0 : répertoire vide
 - Révision i : projet après i propagations
- **svn update** : met à jour les fichiers à partir de la dernière révision du projet

Gestion des révisions

- Gestion des révisions dans le dépôt



Gestion des révisions

- **Numéro de révision et fichiers**
 - identifie la version d'un projet ET NON d'un fichier
 - A chaque fichier est associé le numéro de révision du projet
 - Exemple : Révision 5 du fichier Plateau.cpp
 - état de Plateau.cpp dans la révision 5 du projet
 - n'est pas la 5eme version du fichier Plateau.cpp
 - est identique à la révision 10 de Plateau.cpp si Plateau.cpp n'a pas changé entre les 5ème et 10ème révisions.

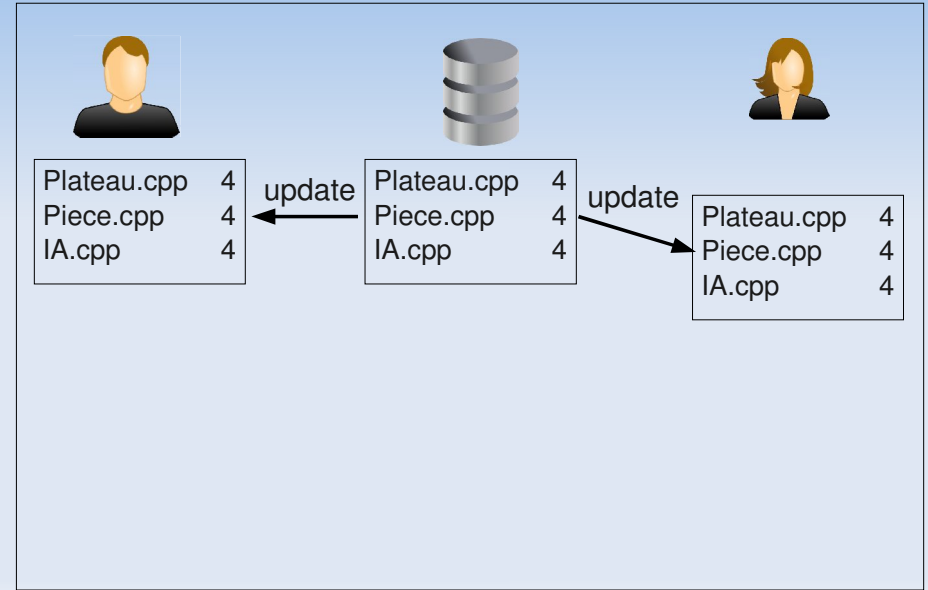
Gestion des révisions

- Propagation par **svn commit** :
 - propage uniquement les éléments modifiés
 - maj des num de révision de ces fichiers uniquement
- Mise à jour par **svn update** :
 - maj des éléments modifiés sur le dépôt depuis la dernière mise à jour
 - maj des num de révision de tous les fichiers
- fichiers d'une même copie de travail peuvent être associés à des num de révision différents

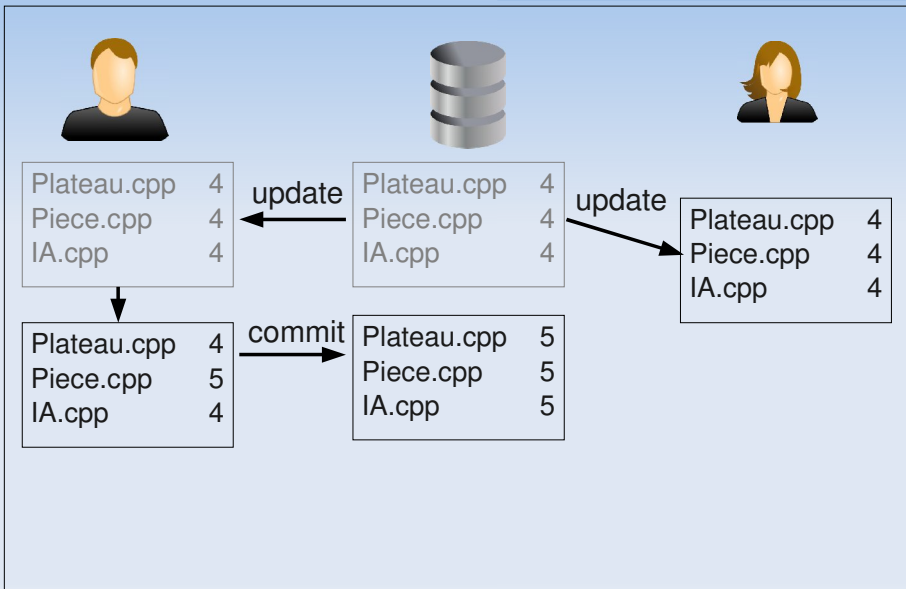
Gestion des versions



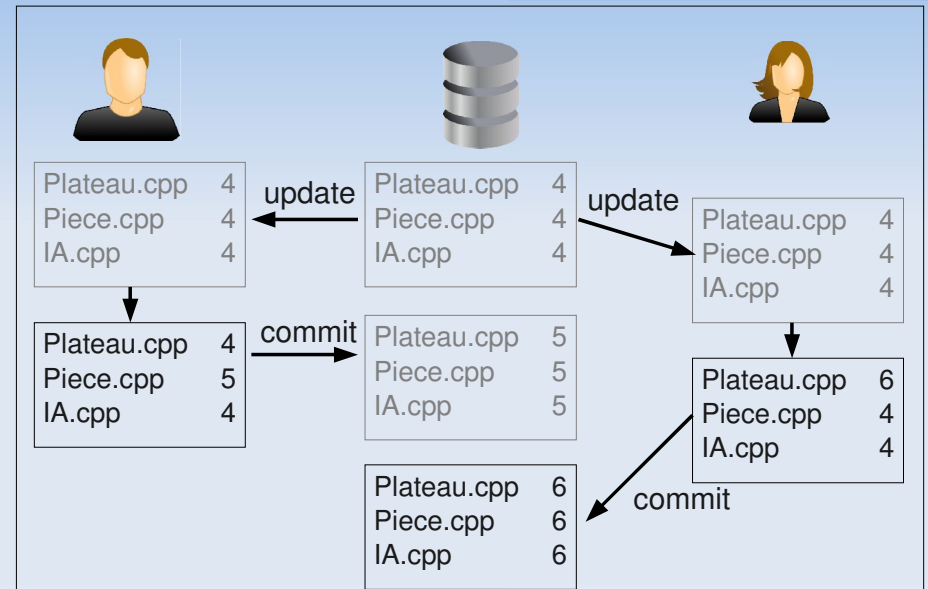
Gestion des versions



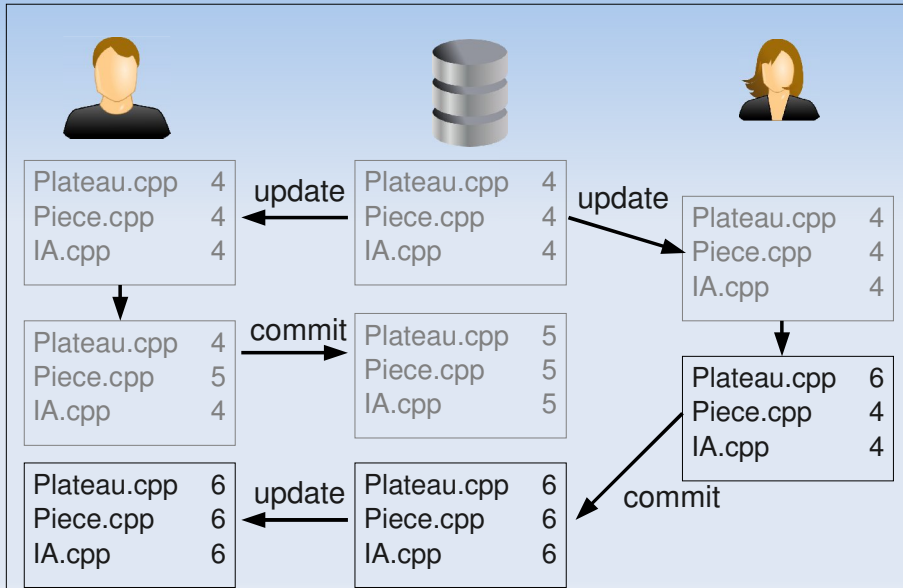
Gestion des versions



Gestion des versions



Gestion des versions



Gestion des révisions

- Pour tout fichier, deux éléments dans «.svn» :
 - sur quelle révision de projet est basé le fichier
 - quand la copie locale a été mise à jour pour la dernière fois depuis le dépôt
- Dialogue avec dépôt → État de chaque fichier :
 - Inchangé et à jour
 - Modifié localement, et à jour
 - Inchangé et périmé
 - Modifié localement, et périmé

Gestion des révisions

- État du fichier : **inchangé et à jour**
 - Fichier inchangé dans la copie de travail depuis la dernière mise à jour
 - Aucune modification propagée vers le dépôt par un autre utilisateur
 - **svn commit** : aucun effet, aucune modification locale à propager
 - **svn update** : aucun effet, déjà à la dernière version

Gestion des révisions

- État du fichier : **modifié localement et à jour**
 - Fichier modifié localement dans la copie de travail depuis la dernière mise à jour
 - Aucune modification propagée vers le dépôt par un autre utilisateur
 - Il existe des modifications à propager vers le dépôt
 - **svn commit** : va propager les modifications locales
 - **svn update** : aucun effet, déjà à la dernière version

Gestion des révisions

- État du fichier : **inchangé et périmé**
 - Fichier inchangé dans la copie de travail depuis la dernière mise à jour
 - Modifications propagées par un autre utilisateur
→ Le fichier a changé sur le dépôt
 - **svn commit** : aucun effet, aucune modification locale à propager
 - **svn update** : mise à jour du fichier

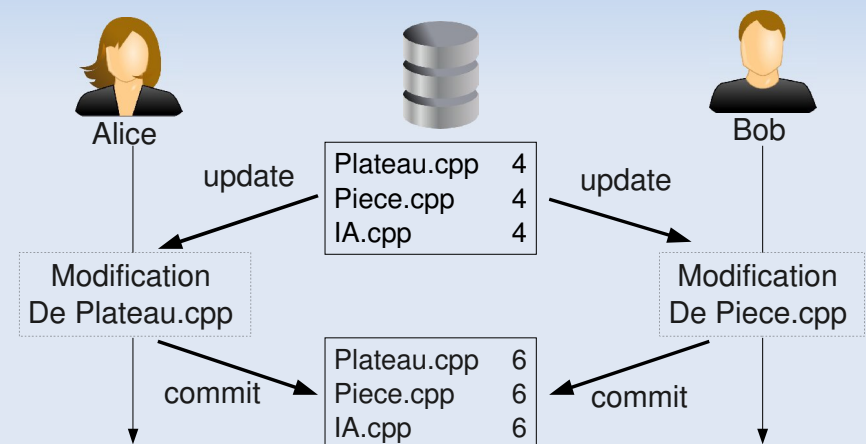
Gestion des révisions

- État du fichier : **modifié localement et périmé**
 - Fichier modifié localement dans la copie de travail depuis la dernière mise à jour
 - Modifications propagées par un autre utilisateur
 - Deux versions concurrentes : locale vs dépôt
 - **svn commit** : échoue, « version périmée». Besoin de faire une mise à jour
 - **svn update** : Subversion tente de fusionner la version locale et celle du dépôt
 - Succès → Modifié localement et à jour
 - Échec : → Conflits à régler par l'utilisateur

Fusion des révisions,
Gestion des conflits

Fusion des révisions, Gestion des conflits

- Chaque utilisateur travaille sur un fichier propre

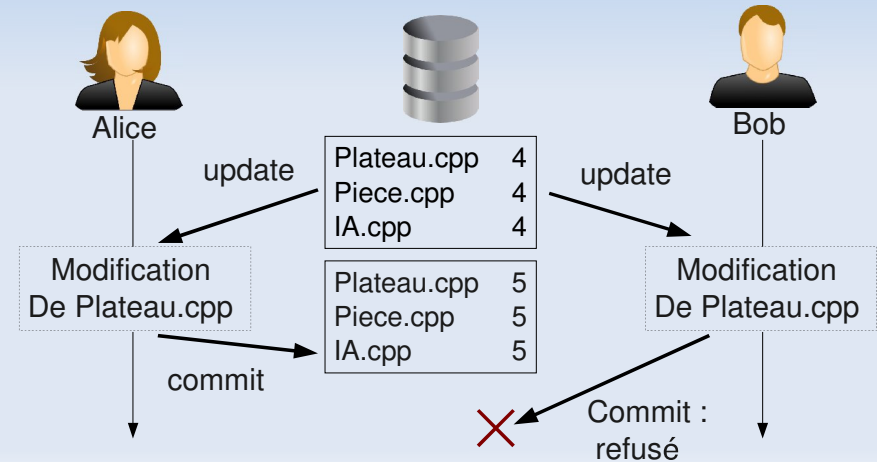


Fusion des révisions, Gestion des conflits

- Chaque utilisateur travaille sur un fichier propre
 - État de chaque fichier avant modification: «inchangé et à jour »
 - État de chaque fichier après modification : « **modifié localement et à jour** »
- **svn commit** : propagation des fichiers modifiés
 - Éditions parallèle sans aucun conflit
 - Propagation transparente: aucun utilisateur ne sait si l'autre a modifié d'autres fichiers
- **svn update** : mise à jour des autres fichiers

Fusion des révisions, Gestion des conflits

- Des utilisateurs travaillent sur le même fichier



Fusion des révisions, Gestion des conflits

- Des utilisateurs travaillent sur le même fichier
 - État de chaque fichier avant modification: «inchangé et à jour »
- Premier utilisateur à propager sa révision :
 - Dans l'exemple précédent : Alice
 - État de Plateau.cpp après modification : « **modifié localement et à jour** »
 - **svn commit** : propagation des fichiers modifiés

Fusion des révisions, Gestion des conflits

- Utilisateurs suivant propageant leur révision
 - Dans l'exemple précédent : Bob
 - État de Plateau.cpp après modification : « **modifié localement et périmé** »
 - **svn commit** : échoue, « version périmée »
 - Besoin de faire une mise à jour avant propagation
- **svn update** : Subversion tente de fusionner la version locale modifiée et celle du dépôt

Fusion des révisions, Gestion des conflits

- Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```



Fusion des révisions, Gestion des conflits

- Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```



← update



update →



Fusion des révisions, Gestion des conflits

- Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
  int somme;  
  somme =a+b;  
  return somme;  
}  
  
int main() {  
  // a faire  
}
```

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```



Modification
Du fichier



Fusion des révisions, Gestion des conflits

- Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
  int somme;  
  somme =a+b;  
  return somme;  
}  
  
int main() {  
  // a faire  
}
```

```
int addition (int a, int b) {  
  int somme;  
  somme =a+b;  
  return somme;  
}  
  
int main() {  
  // a faire  
}
```

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```



→ commit



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {  
    // a faire  
}
```

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {  
    // a faire  
}
```

```
int addition (int a, int b) {  
    // a faire  
}
```

```
int main() {  
    int op1, op2;  
    op1 = addition(1,5);  
    op2 = addition(op1,3);  
    printf("%i", op2);  
}
```



Modification
Du fichier



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {  
    // a faire  
}
```

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {  
    // a faire  
}
```

```
int addition (int a, int b) {  
    // a faire  
}
```

```
int main() {  
    int op1, op2;  
    op1 = addition(1,5);  
    op2 = addition(op1,3);  
    printf("%i", op2);  
}
```



← commit



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {  
    // a faire  
}
```

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {
```

```
int addition (int a, int b) {  
    // a faire  
}
```

```
int main() {  
    int op1, op2;  
    op1 = addition(1,5);  
    op2 = addition(op1,3);  
    printf("%i", op2);  
}
```

Echec de commit :
La version de travail avant modification n'était pas la dernière du dépôt. Faire **svn update**



~~← commit~~



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers sans conflits

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {  
    // a faire  
}
```

```
int addition (int a, int b) {  
    int somme;  
    somme =a+b;  
    return somme;  
}
```

```
int main() {  
    // a faire  
}
```

```
int addition (int a, int b) {  
    // a faire  
}
```

```
int main() {  
    int op1, op2;  
    op1 = addition(1,5);  
    op2 = addition(op1,3);  
    printf("%i", op2);  
}
```

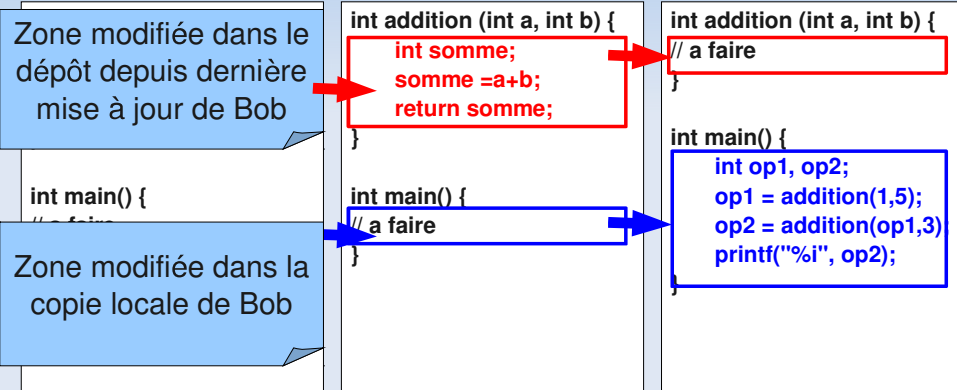


→ update



Fusion des révisions, Gestion des conflits

Fusion de fichiers sans conflits

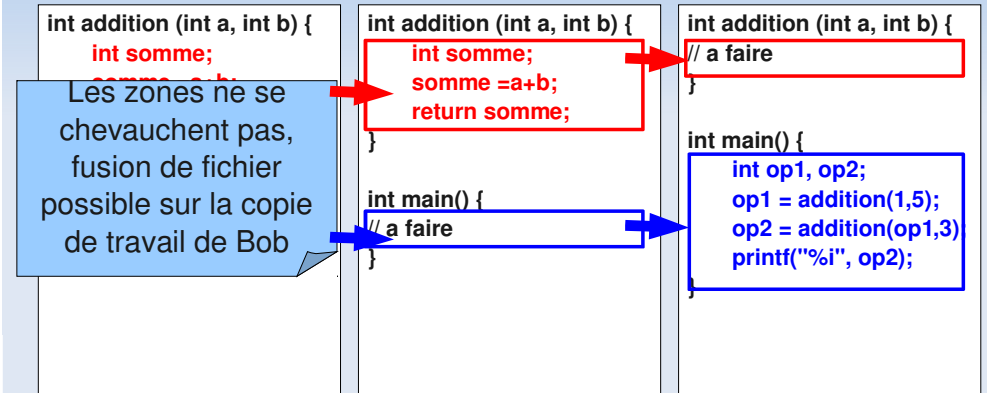


update



Fusion des révisions, Gestion des conflits

Fusion de fichiers sans conflits

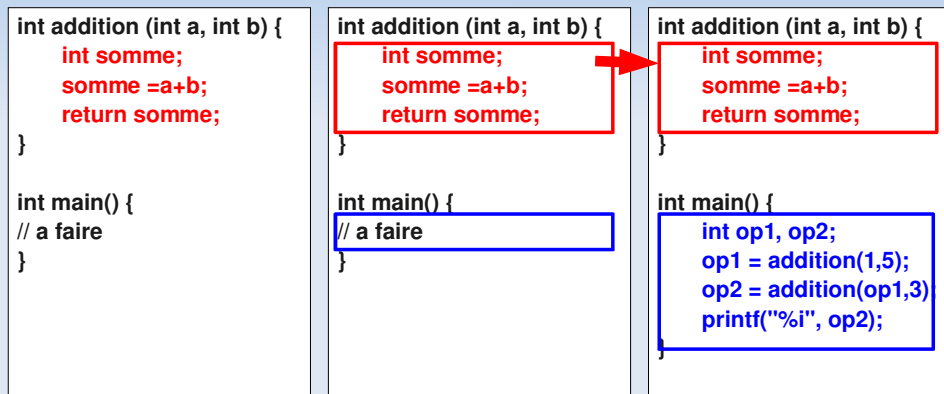


update



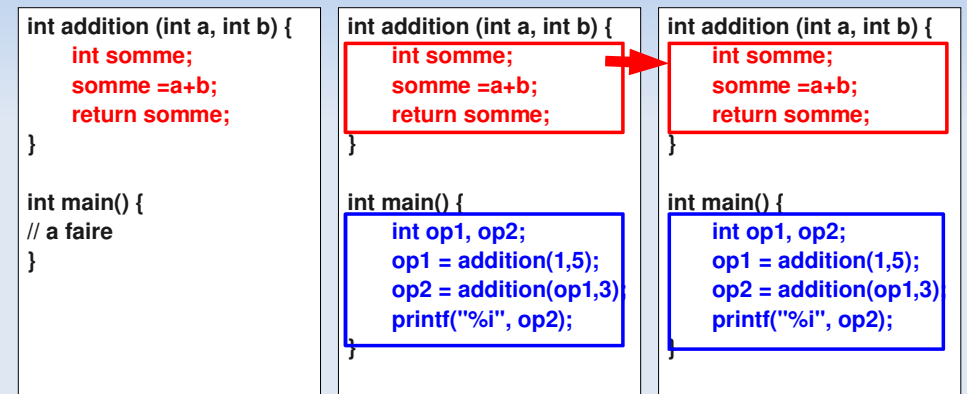
Fusion des révisions, Gestion des conflits

Fusion de fichiers sans conflits



Fusion des révisions, Gestion des conflits

Fusion de fichiers sans conflits



commit



Fusion des révisions, Gestion des conflits

- Fusion de fichiers avec conflits

Fusion des révisions, Gestion des conflits

- Fusion de fichiers avec conflits

```
int addition (int a, int b) {  
  // a faire  
}  
  
int main() {  
  // a faire  
}
```



Fusion des révisions, Gestion des conflits

- Fusion de fichiers avec conflits

<pre>int addition (int a, int b) { // a faire } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { // a faire } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { // a faire } int main() { // a faire }</pre>
--	--	--



← update



update →



Fusion des révisions, Gestion des conflits

- Fusion de fichiers avec conflits

<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { // a faire } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { // a faire } int main() { // a faire }</pre>
---	--	--



Modification
Du fichier



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers avec conflits

<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { // a faire } int main() { // a faire }</pre>
--	--	--



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers avec conflits

<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { //retourner a+b return a+b; } int main() { // a faire }</pre>
--	--	---



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers avec conflits

<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { //retourner a+b return a+b; } int main() { // a faire }</pre>
--	--	---

Echec de commit :
La version de travail avant modification n'était pas la dernière du dépôt. Faire **svn update**



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers avec conflits

<pre>int main() { // a faire }</pre>	<pre>int addition (int a, int b) { int somme; somme =a+b; return somme; } int main() { // a faire }</pre>	<pre>int addition (int a, int b) { //retourner a+b return a+b; } int main() { // a faire }</pre>
--	--	---

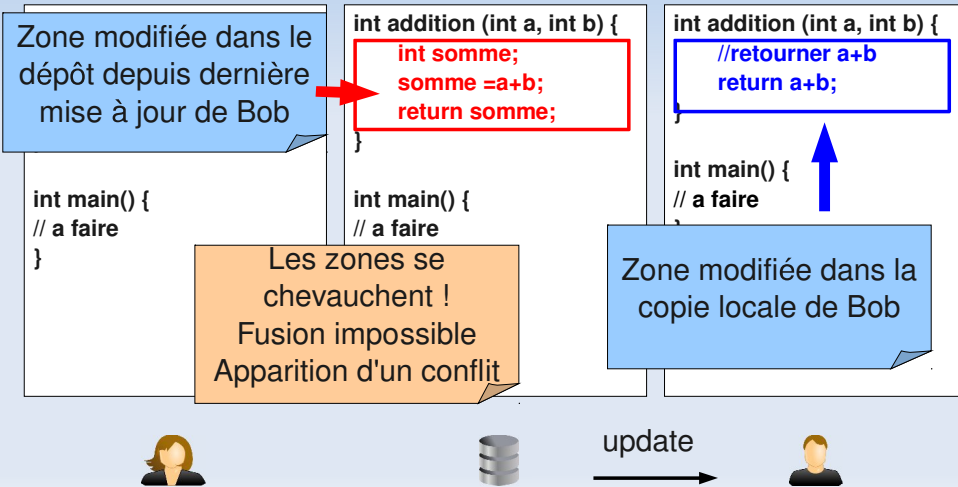
Zone modifiée dans le dépôt depuis dernière mise à jour de Bob

Zone modifiée dans la copie locale de Bob



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers avec conflits



Fusion des révisions, Gestion des conflits

▪ Fusion de fichiers avec conflits

- Subversion ne parvient pas à fusionner les fichiers
- Présence d'un conflit
- Situation la plus problématique
- Intervention de l'utilisateur nécessaire (ici Bob)
 - Éditer sa copie
 - Déterminer les éléments à garder / supprimer
 - Indiquer que le conflit a été résolu
 - Propager sa version

Fusion des révisions, Gestion des conflits

▪ Que voit Bob lorsqu'il exécute `svn update` ?

```
$ svn update
```

```
Conflit découvert dans 'fichier.cpp'.
```

```
Sélectionner : (p) report, (df) diff entier, (e)  
édite, (mc) mes conflits, (tc) autres conflits,  
(s) affiche toutes les options :
```

▪ Subversion : attente de décision ...

Fusion des révisions, Gestion des conflits

▪ Affichage de toutes les options : 's'

```
(e) édite - résout manuellement le conflit avec  
un éditeur  
(df) diff complet - montre toutes les différences  
du fichier fusionné  
(r) résolu - utilise la version fusionnée  
(dc) affiche conflits - affiche tous les conflits  
(ignore version fusionnée)  
(mc) mes conflits - accepte ma version pour tous  
les conflits  
(tc) autres conflits - accepte l'autre version  
pour tous les conflits
```

Fusion des révisions, Gestion des conflits

- Affichage de toutes les options : 's'

```
(mf) mien complet - utilise ma version (ignore
les autres éditions)
(tf) autre complet - prends la version du
dépôt (perds mes éditions)
(p) report - marque ce conflit pour résolution
ultérieure
(l) lance - utilise un outil externe pour
résoudre le conflit
(s) aide - affiche cette liste
```

Fusion des révisions, Gestion des conflits

- Report du conflit pour résolution ultérieure : 'p'
 - Subversion fusionne les deux versions en incluant les deux modifications
 - Utilise les balises suivantes sur la zone en conflit :

```
<<<<<<< .mine
(version locale)
...
=====
(version du dépôt)
...
>>>>>>> .r(num révision dépôt)
```

- Garde en mémoire la présence d'un conflit

Fusion des révisions, Gestion des conflits

- Report du conflit pour résolution ultérieure : 'p'
 - Version copie de travail de Bob :

```
int addition (int a, int b) {
<<<<<<< .mine
//retourner a+b
return a+b;
=====
int somme;
somme =a+b;
return somme;
>>>>>>> .r2
}

int main() {...
```

Fusion des révisions, Gestion des conflits

- Résolution d'un conflit reporté
 - Édition du fichier : suppression des éléments indésirables
 - indiquer à Subversion que le conflit est résolu: **svn resolved fichier**
 - propagation du fichier : **svn commit ...**
- Omission de **svn resolved** avant **svn commit** :
 - Échec de la propagation : le fichier demeure toujours en conflit pour Subversion

Fusion des révisions, Gestion des conflits

- Autres alternatives :
 - Visualiser les différences : df, dc
 - Privilégier ses modifications : mc, mf
 - Privilégier les modifications des autres : tc, tf

Ajout / suppression de fichiers à un dépôt
Renommage et déplacement

Ajout de nouveaux fichiers

- Notion de fichier versionnés / non versionnés
 - Un fichier contenu dans le répertoire de copie de travail n'appartient pas forcément au projet !
 - fichiers temporaires : *.c~, *.tmp, etc ...
 - fichiers objets : *.o
 - fichiers pas encore ajoutés au projet
 - exécutables : *.exe
 - autres fichiers
 - Fichier versionné : appartient au projet
 - Fichier non versionné : n'appartient pas au projet
 - Propagation : fichiers « versionnés » uniquement

Ajout de nouveaux fichiers

- Modification de fichiers existants : OK
- Besoin d'ajouter de nouveaux fichiers
- Deux méthodes distinctes
 - **svn import**
 - **svn add**

Ajout de nouveaux fichiers

- Ajouter une arborescence au dépôt
svn import
 - Deux paramètres :
 - Arborescence à importer
 - URL du dépôt
 - Copie une arborescence entière vers le dépôt
 - Message décrivant l'ajout : **-m**
 - L'arborescence importée :
 - n'est pas nécessairement une copie de travail
 - n'est pas nécessairement versionnée
 - Utilisé pour ajout de projets pas encore versionnés

Ajout de nouveaux fichiers

- Exemple :

```
$ svn import jeuEchec svn://monServ.org/jeuEchec

Ajout   jeuEchec/
Ajout   jeuEchec/plateau.cpp
Ajout   jeuEchec/Piece.cpp
Ajout   jeuEchec/plugin/
Ajout   jeuEchec/plugin/IA.cpp
Ajout   jeuEchec/jeuEchec/GUI.cpp

Révision 1 propagée
```

Ajout de nouveaux fichiers

- Passer un fichier non versionné → versionné :
svn add
 - Paramètres : liste des fichiers à ajouter
 - Ajout récursif sur les répertoires
 - Portée de l'ajout limitée à la copie de travail !
 - Propager cet ajout au dépôt : **svn commit ...**
 - Fonctionne également sur les répertoires

Ajout de nouveaux fichiers

- Exemple :

```
$ (positionneemnt dans un répertoire versionné)

$ echo "include <stdio.h>" > Plateau.cpp

$ svn add Plateau.cpp
A Plateau.cpp

$ svn commit -m "création de Plateau.cpp"
Ajout   Plateau.cpp
Révision 1 propagée
```


Ajouts de nouveaux répertoires

- Créer un nouveau répertoire

- **Méthode 1 :**

```
mkdir mon_rep  
svn add mon_rep  
svn commit -m « nouveau repertoire »
```

- **Methode 2 :**

```
svn mkdir mon_rep  
svn commit -m « nouveau repertoire »
```

Suppression de fichiers

- Que se passe-t'il si je supprime manuellement un fichier versionné dans la copie de travail?
 - Subversion ne sait pas qu'un fichier a été supprimé
 - Fichier toujours référencé dans le répertoire **.svn**
- **svn commit** : échoue → le fichier est manquant
- **svn update** : recrée le fichier supprimé à partir de la dernière version propagée sur le dépôt

Suppression de fichiers

- Supprimer des fichiers d'un projet :
svn delete

- Deux configurations possible :

- suppression du projet ET de la copie de travail
→ configuration par défaut !
- suppression du projet seulement, conservation du fichier local : versionné → non versionné
→ ajout du flag **--keep-local** à la l.d.c.

- Commande à manipuler avec précaution!

Suppression de fichiers

- Exemple :

```
•$ svn delete Plateau.cpp  
// Plateau.cpp retiré des fichiers versionnés  
  
$ svn commit -m "suppression Plateau.cpp"  
// Plateau.cpp supprimé du dépôt & copie de travail
```

```
$ svn delete Plateau.cpp --keep-local  
// Plateau.cpp retiré des fichiers versionnés  
  
$ svn commit -m "suppression Plateau.cpp"  
// Plateau.cpp supprimé du dépôt
```

Suppression de fichiers

- Restrictions de suppression (1 / 2) :
 - Fichier versionné qui va être supprimé :
→ ne doit pas avoir été modifié localement
 - si modification locale, suppression non autorisée **svn delete fichier** échoue
 - solution :
 - propager les changements : **svn commit ...**
 - Supprimer localement : **svn delete fichier**
 - propager à nouveau : **svn commit ...**

Renommage et déplacement de fichiers sur la copie de travail

- Tout renommage ou déplacement de fichier sur la copie de travail :
 - N'est pas forcément détectable par Subversion
 - Doit en conséquence être signalée à Subversion, de la même façon que pour supprimer un fichier
 - Autrement le fichier ne sera pas trouvé et sera considéré comme manquant (comportement id)
- Commandes référence :
 - Déplacer un fichier : **svn move source cible**
 - Renommer un fichier : **svn rename source cible**

Quelques sous-commandes utiles

Quelques sous-commandes utiles

- Visualiser les changements avant publication **svn status**
 - Liste des éléments qui vont être propagés par le prochain appel à svn commit
 - Éléments ajoutés
 - Éléments supprimés
 - Éléments modifiés
 - Affichage de la révision de travail
 - Affichage de la dernière révision propagée + auteur
 - Présence et état de verrous

Quelques sous-commandes utiles

- Afficher l'historique d'un fichier / répertoire
svn log
 - Liste des gens qui ont modifié le fichier
 - Numéro de chaque révision
 - Message associé a la propagation
 - Historique affiché en ordre chronologique inversé

Quelques sous-commandes utiles

- Visualiser les modifications par fichier
svn diff
 - Comparaison entre deux révisions d'un fichier
 - Affichage selon format unifié diff (commande unix)
 - Les lignes ajoutées sont précédées par un +
 - Les lignes supprimées sont précédées par un -
 - Utilise un cache sur les fichiers présent dans **.svn**

Quelques sous-commandes utiles

- Lister les fichiers d'un dépôt sans les charger
svn list
 - Paramètre : url du dépôt
 - Sans paramètre : liste des fichiers du répertoire local si ce dernier est versionné
 - Affichage plus détaillé : option `--verbose`

Quelques sous-commandes utiles

- Annuler des changements sur copie de travail
svn revert
 - Restaurer l'état initial d'un fichier
 - Annuler les modifications locales
 - Paramètre : fichier / répertoire a restaurer

Verrouillage / Déverrouillage

Verrouillage / Déverrouillage

- Poser un verrou pour édition exclusive
svn lock
 - Paramètre obligatoire :
 - Fichier à verrouiller
 - Paramètre recommandé :
 - Message stipulant la raison du verrou : option -m
 - Le fichier à verrouiller doit être à jour !
 - Pas de copie obsolète (faire svn update autrement)

```
$ svn lock Plateau.cpp -m "je travaille seul dessus"  
'Plateau.cpp' verrouillé par l'utilisateur 'Benoit'
```

Verrouillage / Déverrouillage

Verrouillage / Déverrouillage

- Vérification de la présence d'un verrou
 - **svn status**
 - Affiche un K : lockEd
 - **svn info**
 - Paramètre : nom du fichier
 - Informations détaillées sur le verrou
 - Detenteur du verrou
 - Date de création
 - Commentaire du verrou

- Déverrouillage automatique
 - Un utilisateur verrouille un fichier
 - il modifie ce dernier
 - lors de la propagation du fichier par **svn commit** :
 - le verrou est automatiquement supprimé !
 - Les verrous posés sur les autres fichiers par le même utilisateur sont également supprimés, même si les fichiers n'ont pas été modifiés!
 - évite les verrous laissés négligemment
 - Dissuade l'utilisateur de garder des verrous trop longtemps

Verrouillage / Déverrouillage

- Déverrouillage manuel
svn unlock
 - Supprime un verrou précédemment posé
 - Détendeur du verrou uniquement ?
- Cassage de verrou
 - L'administrateur peut casser le verrou
 - Par défaut, tout utilisateur peut également casser un verrou : ajout de l'option **--force**
 - Permet de supprimer les verrous oubliés

Résolution de problèmes

Verrouillage / Déverrouillage

- Vol de verrou
 - Alice a verrouillé fichier.doc avec **svn lock**
 - Bob veut casser ce verrou, et le verrouiller avec son propre verrou :
 - **svn unlock --force** fichier.doc
 - **svn lock** fichier.doc
 - Opération réalisable en une seule commande :
 - **svn lock --force** fichier.doc

Résolution de problèmes

- Le répertoire **.svn** a été accidentellement supprimé :
 - Sauvegarder les fichiers dont les changements n'ont pas été propagés
 - Supprimer tout le répertoire qui devait contenir ce répertoire avec les outils de suppression du système et pas avec **svn delete**
 - Effectuer un **svn update**
 - Cette commande recréera le répertoire **.svn** et le répertoire précédemment supprimé à partir de la dernière version propagée sur le serveur