TP n°1
Gestion de version

Introduction

RCS (Révision Control System) est un système constitué d'un ensemble d'outils qui, sous Linux[1], permet de gérer les différentes versions d'un code, d'un document, etc. Un texte expliquant les modifications peut (doit !) être saisi par l'utilisateur. L'utilisation repose sur 3 commandes principales : `rcs`, `ci` et `co`.

Mise en oeuvre de base

- dans le répertoire de travail, créer un répertoire RCS
- lorsque l'on crée un nouveau fichier où que l'on a modifié un fichier existant, une fois celui-ci sauvé, faire un
  ```
  ci -u fichier
  ```
  Ceci archive la nouvelle version et garde une copie courante du fichier dans le répertoire de travail.
- Lorsque l'on souhaite modifier un fichier, faire un
  ```
  co -l fichier
  ```
  Ceci verrouille l'archive du fichier empéchant à tout autre utilisateur de le modifier. La dernière version est copiée dans le répertoire de travail en lecture/écriture.

Utilisation avancée

- Pour changer le numéro de *release*, faire un `ci -u -rNUM fichier`.
- Pour récupérer une ancienne version, faire un `co -l -rNUM fichier`.
- De nouvelles branches de développement peuvent être créées grâce à un :
  ```
  ci -u -rNUM.1 fichier
  ```
- Pour connaître les différences entre versions, voir `rcsdiff`.
- Pour regrouper des versions, voir `rcsmerge`.

Application

1. Ecrire une classe Java `Point` comportant deux attributs de type entier `x` et `y` avec les accesseurs en lecture/écriture pour chacun d'eux, un constructeur par défaut, un à deux paramètres entier, et un constructeur de copie. Ecrire aussi la méthode `toString()`. Archiver le fichier comme première version (1.1).
2. Modifier la classe `Point` pour qu'un attribut réel `couleur` soit ajouté. Le constructeur par défaut reste, le constructeur de copie également, mais le constructeur prenant deux paramètres est remplacé par un constructeur à trois paramètres. Des accesseurs à l'attribut couleur sont ajoutés. La méthode `toString()` est modifiée en conséquence. Archiver la nouvelle version (1.2).
3. Ajouter à la version précédente un `main()` de test et archiver cette nouvelle version comme *release* 2.
4. Modifier la version 1.1 pour ajouter un attribut entier `z`. Rajouter les accesseurs à `z`, la méthode `toString()`, modifier le constructeur à 2 paramètres pour qu'il en prenne 3 et archiver le nouveau fichier comme nouvelle branche (1.1.1.1).
5. Modifier la version 1.1.1.1 pour ajouter un paramètre réel `couleur`. Archiver la nouvelle version (1.1.1.2).
6. Ajouter un main() de test à la version 1.1.1.2. L'archiver comme *release* 3.
7. Afficher l'historique des évolutions de la classe Point ainsi que le bilan des différences entre les *release* 2 et 3.

Faciliter l'utilisation

1. Ecrire un programme `edite` qui prend en paramètre un nom de fichier et un éventuel numéro de version. `edite` récupère la version correspondante, et lance emacs sur le fichier en question.
2. Ecrire un programme `archive` qui prend en paramètre un nom de fichier et un éventuel numéro de version. `archive` sauve le fichier dans le répertoire RCS.

Pour aller plus loin

man rcsintro, rcs, ci, co, rcsdiff, rcsmerge, rlog

---

1 Sous d'autres Unix, on trouvera SCCS qui fonctionne de façon similaire