Le MINI-HOWTO RCS

Robert Kiesling
Traduction Jean-Albert Ferrez, <Jean-Albert.Ferrez@epfl.ch> V1.4 1997/08/14

Ce document couvre les bases de l'installation et de l'utilisation de RCS, le système de contrôle de révisions (*Revision Control System*) de GNU, sous Linux. Il couvre également l'installation des utilitaires `diff(1)` et `diff3(1)` qui sont nécessaires au fonctionnement de RCS. Ce document peut être reproduit librement, dans sa totalité ou en partie, pour autant que tout usage de ce document soit conforme à la notice générale sur le copyright de la série des Howto du *Linux Documentation Project*. Référez-vous au fichier COPYRIGHT pour plus de détails. Envoyez toute plainte, suggestion, correction et autre à *kiesling@terracom.net* , pour que je puisse maintenir ce document aussi complet et à jour que possible.

Contents

1 Vue d'ensemble de RCS. **1**

2 Prérequis **2**

3 Compiler RCS **2**

4 Créer et maintenir des archives **3**

5 `ci(1)` et `co(1)`. **3**

6 Historique des révisions **4**

7 Inclure des données RCS dans les fichiers **4**

8 RCS et le contrôle de versions dans `emacs(1)` **4**

1 Vue d'ensemble de RCS.

RCS, le système de contrôle de révisions, est un ensemble de programmes qui suivent les changements dans des fichiers textes et contrôlent les accès concurrents dans le cadre d'un travail collectif. Il est généralement utilisé pour maintenir des collections de codes sources. Il est également adapté au suivi des fichiers de documentation.

RCS a été écrit par Walter F. Tichy et Paul Eggert. La dernière version ayant été portée sous Linux est la 5.7. Il existe également une version semi officielle multitâche (*threaded*). La plupart des informations de ce Howto proviennent des pages de manuel de RCS.

RCS comprend le programme `rcs(1)` qui contrôle les attributs de l'archive RCS, `ci(1)` et `co(1)`, qui enregistrent un fichier dans l'archive (*check in*) et extraient des fichiers de l'archive (*check out*), `ident(1)`, qui recherche un mot-clé dans une archive RCS, `rcsclean(1)`, qui fait le ménage en éliminant les fichiers inchangés et sur lesquels personne ne travaille, `rcsdiff(1)`, qui exécute `diff(1)` pour comparer les révisions, `rcsmerge(1)`, qui fusionne deux branches de développement en un seul fichier, et `rlog(1)`, qui affiche l'historique des modifications.

Les fichiers archivés avec RCS peuvent être du texte d'un format quelconque, ou des fichiers binaires si le `diff` utilisé pour générer les changements supporte les données sur 8 bits. Les fichiers peuvent contenir un texte de description pour faciliter le suivi par `ident`. RCS utilise les programmes `diff(1)` and `diff3(1)` pour générer les modifications entre les diverses révisions. Une archive RCS consiste en la révision initiale - la version 1.1 - et une série de modifications, une pour chaque révision. Chaque fois qu'un fichier est extrait de l'archive à l'aide de `co(1)`, édité, puis à nouveau enregistré dans l'archive avec `ci(1)`, le numéro de version est incrémenté, par exemple 1.2, 1.3, 1.4, etc. pour les révisions successives.

Les fichiers archives eux-mêmes se trouvent généralement dans un sous-répertoire `./RCS`, bien que RCS ait d'autres options pour le stockage des archives.

Pour une vue d'ensemble de RCS, voyez également la page de manuel de `rcsintro(1)`.

2 Prérequis

RCS a besoin de `diff(1)` et de `diff3(1)` pour générer les modifications entre les révisions. La suite d'utilitaires diff doit être installée sur votre système; RCS s'assure de sa présence lors de l'installation.

Des binaires pré-compilés des diffutils sont disponibles à l'adresse :

```
ftp://sunsite.unc.edu/pub/Linux/utils/text/diffutils-2.6.bin.ELF.tar.gz
```

ainsi que sur les sites miroirs. Si vous devez les compiler, les sources se trouvent à :

```
ftp://prep.ai.mit.edu/pub/gnu/diffutils-2.7.tar.gz
```

ainsi que sur les sites miroirs.

NdT: En France, on trouve les diffutils sur `ftp.lip6.fr` aux adresses suivantes :

```
ftp://ftp.lip6.fr/pub/linux/sunsite/utils/text/diffutils-2.6.bin.ELF.tar.gz
ftp://ftp.lip6.fr/pub/gnu/diffutils-2.7.tar.gz
```

Les librairies ELF doivent également être installées sur votre système pour utiliser les binaires pré-compilés. Reportez-vous au ELF-Howto pour plus de détails.

3 Compiler RCS

Procurez-vous les sources de RCS version 5.7, disponibles à :

```
ftp://sunsite.unc.edu/pub/Linux/devel/vc/rcs-5.7.src.tar.gz
```

ainsi que sur les sites miroirs, en France à :

```
ftp://ftp.lip6.fr/pub/linux/sunsite/devel/vc/rcs-5.7.src.tar.gz
```

Après avoir extrait l'archive, il faut configurer RCS pour votre système. Ceci se fait à l'aide du script `configure` dans le répertoire source, qu'il faut exécuter en premier. Ceci va générer une `Makefile` ainsi que le `conf.sh` approprié pour votre système. Vous pouvez ensuite faire

```
make install
```

ce qui va créer les binaires. Vous devrez peut-être devenir root à un certain moment pour installer les binaires dans les bons répertoires.

4 Créer et maintenir des archives

Le programme `rcs(1)` s'occupe de créer des archives et de modifier leurs attributs. Les options de `rcs(1)` sont données dans la page de manuel `rcs(1)`.

La manière la plus facile de créer une archive est d'exécuter `mkdir RCS` dans le répertoire courant, puis d'initialiser l'archive avec la commande

```
rcs -i nom_du_fichier_de_travail
```

Ceci va créer une archive nommée `./RCS/nom_du_fichier_de_travail,v` et réclamer un texte décrivant l'archive, mais ne dépose aucune révision dans l'archive. Vous pouvez enclencher et déclencher le blocage strict avec les commandes

```
rcs -L nom_du_fichier_de_travail
```

et

```
rcs -U nom_du_fichier_de_travail
```

respectivement. Il y a d'autres options pour contrôler l'accès à l'archive, fixer son format et ses numéros de révisions; tout est expliqué dans la page de manuel `rcs(1)`.

5 `ci(1)` et `co(1)`.

`ci(1)` et `co(1)` sont les commandes utilisées pour enregistrer un fichier dans son archive et l'en extraire. La commande `ci(1)` peut également être utilisée pour les deux opérations. Dans leur forme la plus simple, `ci(1)` and `co(1)` ne nécessitent que le nom du fichier de travail.

```
ci nom_du_fichier_de_travail
```

et

```
co nom_du_fichier_de_travail
```

La forme suivante

```
ci -l nom_du_fichier_de_travail
```

enregistre le ficher en mode bloqué, et

```
co -l nom_du_fichier_de_travail
```

est exécuté automatiquement. C'est-à-dire, `ci -l` extrait à nouveau le fichier et le bloque.

```
ci -u nom_du_fichier_de_travail
```

enregistre le fichier dans l'archive et l'extrait à nouveau sans le bloquer. Dans tous les cas, l'utilisateur se voit demander un message pour l'historique.

`ci(1)` crée automatiquement une archive RCS si elle n'existe pas déjà.

Si vous ne spécifiez pas de numéro de révision, `ci(1)` incrémente le dernier numéro présent dans l'archive et y ajoute la version actuelle. Si vous spécifiez un numéro de révision dans une branche existante, il doit être supérieur aux numéros existants. `ci(1)` crée une nouvelle branche si vous spécifiez un numéro dans une branche qui n'existe pas. Référez-vous à la page de manuel de `ci(1)` et `co(1)` pour plus de détails.

`ci(1)` et `co(1)` ont de nombreuses options, en mode interactif ou non. De nouveau, référez-vous aux pages de manuel de `ci(1)` et `co(1)` pour plus de détails.

6 Historique des révisions

La commande `rlog(1)` donne des informations sur une archive ainsi que les commentaires associés à chacune des révisions qu'elle contient. Par exemple :

```
rlog nom_du_fichier_de_travail
```

va afficher la liste des révisions du fichier, avec pour chacune la date, le `userid` de l'auteur et la personne qui a bloqué le fichier. Vous pouvez spécifier les attributs que vous désirez voir.

7 Inclure des données RCS dans les fichiers

`co(1)` tient à jour une liste de mots-clés de l'archive RCS lorsque le fichier de travail est extrait. Le mot-clé `$Id$` dans un document sera remplacé par une chaîne contenant le nom du fichier, le numéro de révision, la date d'extraction, l'auteur, l'état de l'archive et, le cas échéant, la personne qui a bloqué le fichier. Le mot clé `$Log$` est lui remplacé par l'historique du fichier.

Ces mots-clés ainsi que d'autres peuvent être utilisés comme clés de recherche dans une archive RCS. Référez-vous à la page de manuel de `ident(1)` pour plus de détails.

8 RCS et le contrôle de versions dans `emacs(1)`

Le contrôle de versions dans `emacs(1)` fonctionne comme une interface à RCS. Ce qui suit s'applique à la version 19.34 de GNU Emacs qui est livrée avec la majorité des distributions Linux. Si l'on édite dans `emacs(1)` un fichier qui est sous le contrôle de RCS, la commande `vc-toggle-read-only` (associée par défaut à `C-x C-q`) va enregistrer le fichier dans le système de contrôle de version d'emacs, puis dans RCS. emacs ouvre un tampon (*buffer*) dans lequel il est possible de saisir un message pour l'historique. Une fois ce message saisi, `C-c C-c` termine l'édition et procède à l'enregistrement de la nouvelle révision dans l'archive RCS.

Si vous avez opté pour un blocage strict, vous devez bloquer à nouveau le fichier pour l'éditer dans `emacs(1)`. Vous pouvez extraire le fichier à l'aide de la commande `%` dans le mode *buffer-menu*.

Pour plus d'informations, consultez le manuel de GNU Emacs ainsi que les pages info.