The RCS MINI-HOWTO

Robert Kiesling

v1.4, 14 agosto 1997

Questo documento riguarda l'installazione di base e l'uso di RCS, lo GNU Revision Control System (sistema di controllo delle revisioni di GNU), sotto Linux. Viene trattata anche l'installazione delle utilità `diff(1)` e `diff3(1)`, che sono necessarie per il funzionamento di RCS. Mandate reclami, suggerimenti, errata e qualunque altra cosa a *kiesling@terracom.net* , affinché io possa mantenere questo documento il più completo e aggiornato possibile. Traduzione di Fabrizio Stefani (`fabrizio_stefani@yahoo.it`), 22 luglio 1999.

Indice

1 Panoramica di RCS.

RCS, il Sistema di Controllo delle Revisioni (revision control system), è una raccolta di programmi per tenere traccia dei cambiamenti nei file di testo e controllare gli accessi ai file condivisi in situazioni di lavoro di gruppo. Generalmente viene usato per mantenere i moduli di codice sorgente. Si presta anche a tenere traccia delle revisioni dei file di documentazione.

RCS è stato scritto da Walter F. Tichy e Paul Eggert. L'ultima versione di cui è stato fatto il porting per Linux è RCS Versione 5.7. È disponibile anche una versione semi-ufficiale, che usa i thread. Gran parte delle informazioni contenute in questo HOWTO sono state prese dalle pagine di manuale di RCS.

RCS include il programma `rcs(1)`, che controlla gli attributi dei file archivio di RCS, `ci(1)` e `co(1)`, che effettuano il check in ed il check out dei file dagli archivi RCS, `ident(1)`, che effettua ricerche di identificatori chiave negli archivi RCS, `rcsclean(1)`, un programma per cancellare i file su cui non si sta lavorando o che non sono cambiati, `rcsdiff(1)`, che esegue `diff(1)` per confrontare le revisioni, `rcsmerge(1)`, che fonde due rami (branch) RCS in un singolo file di lavoro, e `rlog(1)`, che stampa i messaggi di log di RCS.

I file archiviati con RCS possono essere testo in qualunque formato, o binari se il programma `diff`, usato per generare i file con i cambiamenti, gestisce dati ad 8 bit. I file possono opzionalmente includere stringhe di identificazione, per aiutare nel tracciamento con `ident(1)`. RCS usa le utilità `diff(1)` e `diff3(3)` per generare i file con le modifiche tra più revisioni. Un archivio RCS è formato dalla revisione iniziale di un file, che è la versione 1.1, e una serie di file con le modifiche, uno per ogni revisione. Ogni volta che un file in un archivio viene estratto con `co(1)`, editato e rimesso di nuovo nell'archivio con `ci(1)`, il numero di versione viene incrementato, per esempio ad 1.2, 1.3, 1.4, e così via per le revisioni successive.

Gli archivi stessi di solito risiedono in una sottodirectory `./RCS`, sebbene RCS abbia anche altre opzioni per immagazzinare gli archivi.

Per una presentazione di RCS vedere la pagina di manuale `rcsintro(1)`.

2 Requisiti di sistema.

RCS ha bisogno di `diff(1)` e `diff3(3)` per generare i file con le differenze di contenuto fra le revisioni (i file diff). La suite di utilità diff deve essere installata nel vostro sistema e, quando installate RCS, il software ne controllerà la presenza.

I binari precompilati delle diffutils sono disponibili presso:

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

e sui suoi siti mirror. Se avete bisogno di compilare `diff(1)` (ed altri), dal sorgente, lo trovate presso:

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

e sui suoi siti mirror.

Dovrete anche avere le librerie ELF installate sul vostro sistema, se volete installare i binari precompilati. Vedere l'ELF-HOWTO per ulteriori dettagli.

3 Compilare RCS dal Sorgente.

Prendete la distribuzione sorgente di RCS Versione 5.7. È disponibile presso

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

e sui suoi mirror. Dopo che avete spacchettato l'archivio nel vostro albero dei sorgenti, dovrete configurare RCS per il vostro sistema. Ciò viene fatto attraverso lo script `configure` nella directory sorgente, che dovete eseguire per primo. Così facendo verranno generati un `Makefile` e l'appropriato `conf.sh` per il vostro sistema. Potete poi battere

```
make install
```

che compilerà i binari. Ad un certo punto potreste dover eseguire un `su` a root affinché i binari possano essere installati nelle giuste directory.

4 Creare e mantenere archivi.

Il programma `rcs(1)` effettua il lavoro di creazione degli archivi e di modifica dei loro attributi. Un riassunto delle opzioni di `rcs(1)` può essere trovato nella pagina di manuale di `rcs(1)`.

Il modo più facile per creare un archivio è di effettuare innanzi tutto un `mkdir RCS`, nella directory corrente, e poi inizializzare l'archivio con il comando

```
rcs -i nome_file_di_lavoro
```

Ciò crea un archivio di nome `./RCS/nome_file_di_lavoro,v` e richiede un messaggio di testo per la descrizione dell'archivio, ma non mette nessuna revisione nell'archivio. Potete abilitare o disabilitare il bloccaggio rigoroso (strict locking) dell'archivio con i comandi

```
rcs -L nome_file_di_lavoro
```

e

```
rcs -U nome_file_di_lavoro
```

rispettivamente. Ci sono altre opzioni, trattate nella pagina di manuale di `rcs(1)`, per controllare l'accesso all'archivio, impostarne il formato e impostare i numeri di revisione.

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

`ci(1)` e `co(1)` sono i comandi usati per effettuare il check in (controllo in inserimento) ed il check out (controllo in estrazione) dei file dai loro archivi RCS. Il comando `ci(1)` può anche essere usato per effettuare sia il check in che il check out da un archivio. Nella loro forma più semplice, `ci(1)` e `co(1)` prendono solo il nome del file di lavoro.

```
ci nome_file_di_lavoro
```

e

```
co nome_file_di_lavoro
```

Il comando nella forma

```
ci -l nome_file_di_lavoro
```

effettua il check in del file abilitandone il blocco (lock enabled) mentre

```
co -l nome_file_di_lavoro
```

è effettuato automaticamente. Cioè, `ci -l` effettua di nuovo il check out del file, con il blocco abilitato.

```
ci -u nome_file_di_lavoro
```

effettua il check in del file nell'archivio ed effettua di nuovo il check out con il blocco disabilitato. In tutti i casi, all'utente viene chiesto un messaggio di log.

`ci(1)` creerà anche un archivio RCS se non ne esiste già uno.

Se non specificate una revisione, `ci(1)` incrementa il numero di versione dell'ultima revisione bloccata nell'archivio ed aggiunge allo stesso il file di lavoro appena rivisto. Se specificate una revisione su un ramo già esistente, essa deve essere più alta dei numeri di revisione esistenti. `ci(1)` creerà anche un nuovo ramo se specificate la revisione di un ramo che non esiste. Per i dettagli, vedere le pagine di manuale di `ci(1)` e `co(1)`.

`ci(1)` e `co(1)` riconoscono varie opzioni per usi interattivi e non. Di nuovo, vedere le pagine di manuale di `ci(1)` e `co(1)` per i dettagli.

6 Storia delle revisioni.

Il programma `rlog(1)` fornisce informazioni sul file archivio e i log di ogni revisione in esso immagazzinata. Un comando come

```
rlog nome_file_di_lavoro
```

stamperà la storia delle revisioni del file, per ogni revisione stamperà la data di creazione e lo `userids` dell'autore, e la persona che ha bloccato il file. Potete specificare gli attributi dell'archivio ed i parametri di revisione da vedere.

7 Includere i dati RCS nei file di lavoro.

`co(1)` mantiene una lista di parole chiave del database RCS che vengono espanse quando viene effettuato il check out nel file di lavoro. La parola chiave `$Id$` in un documento sarà espansa in una stringa che contiene il nome del file, il numero di revisione, la data del chek out, l'autore, lo stato della revisione e chi è che lo ha bloccato, se c'è. Includendo la parola chiave `$Log$` essa verrà espansa nel log della storia delle revisioni del documento.

Questa ed altre parole chiave possono essere usate come criteri di ricerca negli archivi RCS. Vedere la pagina di manuale di `ident(1)` per ulteriori dettagli.

8 Il Controllo della Versione con RCS e `emacs(1)`.

La capacità di controllo della versione di `emacs(1)` funziona come front end per RCS. Questa informazione è specifica per la versione 19.34 dello GNU emacs, che è incluso con le principali distribuzioni di Linux. Quando si edita un file con un `emacs(1)` che è registrato con RCS, il comando `vc-toggle-read-only` (legato a `C-x C-q` per default) controllerà la versione di un file in emacs, e poi in RCS. Emacs aprirà un buffer in cui potrete scrivere un messaggio di log, premere `C-c C-c` per terminare l'input, e procedere con il processo di check in.

Se avete selezionato il bloccaggio rigoroso per il file con RCS, dovete ribloccare il file per poterlo editare con `emacs(1)`. Potete fare il chek out del file, per farne il controllo della versione con emacs, con il comando `%` nel modo buffer-menu.

Per maggiori informazioni, vedere il Manuale dello GNU Emacs e le pagine info di Emacs.