SelfLinux-0.10.0

Werkzeuge von Drittanbietern, die mit CVS zusammen arbeiten

Autor: Karl Fogel ()
Formatierung: Matthias Hagedorn (*matthias.hagedorn@selflinux.org*)

Der folgende Text enthält das Kapitel 10 der deutschen Übersetzung des Buches "Open Source Development with CVS", welche unter der GNU Public License veröffentlicht wurden.

Das SelfLinux-Team

Inhaltsverzeichnis

1 Was sind Werkzeuge von Drittanbietern?

Viele Leute haben Programme geschrieben, die den Funktionsumfang von CVS erweitern. Ich nenne diese **Werkzeuge von Drittanbietern**, da sie auch separat zur CVS-Entwicklung verwaltet werden. Die meisten dieser Programme werden nicht zusammen mit CVS verbreitet, einige allerdings schon. Dieses Kapitel behandelt Werkzeuge von Drittanbietern, die ich nützlich finde, die aber nicht mit CVS vertrieben werden.

Obwohl es einige sehr populäre und weit verbreitete Bedienungsoberflächen für CVS gibt, die nicht für die Kommandozeile oder nicht für Unix sind (Download-Adressen sind in Kapitel 4 aufgelistet), behandelt dieses Kapitel die meisten von ihnen nicht. Ihre Popularität macht es einfach, mehr über sie in Newsgruppen und Mailinglisten zu erfahren. Eine Ausnahme ist das **Emacs pcl-cvs-Interface**, das sehr nützlich, aber oft nicht einfach zu installieren ist. Damit fangen wir an.

2 pcl-cvs: eine Emacs-Bedienungsoberfläche für CVS

Benötigt: Emacs, Elib

URLS:

ftp://rum.cs.yale.edu/pub/monnier/pcl-cvs/

Autoren: *Per Cederqvist* und *Stefan Monnier* (derzeitige Betreuung)

`Pcl-cvs` ist eine von zwei Emacs/CVS-Bedienungsoberflächen. Die andere ist die **native VC**-(Version Control-)Oberfläche, die in Emacs eingebaut ist. Ich bevorzuge `pcl-cvs`, weil es speziell für CVS geschrieben wurde und daher sehr angenehm mit der Art harmoniert, in der CVS arbeitet. VC andererseits wurde entwickelt, um mit mehreren unterschiedlichen Versionskontrollsystemen zusammenzuarbeiten - RCS und SCCS genau wie CVS -, und ist daher nicht wirklich optimal auf CVS abgestimmt. Zum Beispiel präsentiert VC eine dateibasierte anstatt einer verzeichnisbasierten Oberfläche zur Revisionskontrolle.

Die Vorteile von `pcl-cvs` wiegen stark genug, um viele Benutzer zum Download und zur Installation zu bewegen, anstatt VC zu benutzen. Unglücklicherweise hat `pcl-cvs` zwei Nachteile: Es kann etwas problematisch in der Installation sein (ein Großteil dieses Abschnitts befasst sich mit möglichen Installationshürden), und seine neuen Versionen laufen etwas instabil.

Das letzte Problem ist wahrscheinlich eher temporärer Natur, es stellt sich jedoch die Frage, welche Version zu benutzen ist. *Stefan Monnier* hat erst kürzlich die Betreuung des `pcl-cvs` übernommen; die neueste Version, 2.9.6 (erhältlich vom ersten URL in der obigen Liste), lief ein bisschen holprig, als ich sie ausprobierte. Zweifellos werden die Probleme binnen kurzem ausgebügelt sein, aber in der Zwischenzeit möchten Sie möglicherweise eine etwas ältere Version benutzen. Da ich Version 1.0.5 lange Zeit täglich benutzt habe, wobei sie sich sehr gut verhielt, werde ich diese Version hier dokumentieren. Glücklicherweise ändert sich an der Installation von Version zu Version nicht allzu viel. Falls Sie sich also dazu entschließen, `pcl-cvs` zu benutzen, so schlage ich vor, dass Sie auf *Stefan Monniers* Seite nach einer neueren Version als 2.9.6 schauen; falls es eine gibt, versuchen Sie die, bevor Sie ganz auf 1.0.5 zurückgehen.

Sie werden feststellen, dass ich zwei URLs für Version 1.0.5 angegeben habe. Die erste ist *Per Cederqvists* Site, wo er ein Archiv alter `pcl-cvs`-Versionen unterhält. Da ich nicht sicher bin, wie lange dieses Archiv noch Bestand haben wird, mache ich außerdem die Version 1.05 auf ftp.red-bean.com verfügbar.

Obwohl der Rest dieser Anweisungen Beispiele aus einer 1.05-Distribution verwendet, sollten diese ebenfalls problemlos auf neuere Versionen anwendbar sein.

3 Installation von pcl-cvs

Falls Sie sich normalerweise nicht mit Problemen der Emacs-Installation und -Instandhaltung beschäftigen, könnte Ihnen die `pcl-cvs`-Installationsprozedur etwas entmutigend erscheinen. Ein paar Hintergrundinformationen zur Arbeitsweise von Emacs hilft dem vielleicht etwas ab.

Die meisten abstrakteren Funktionen von Emacs sind in einer Programmiersprache namens **Emacs Lisp** geschrieben (Emacs selbst ist praktisch ein Interpreter für diese Programmiersprache). Die Leute fügen Emacs neue Funktionen hinzu, indem sie Dateien mit Emacs Lisp-Programmcode verbreiten. `Pcl-cvs` ist in dieser Programmiersprache geschrieben und benötigt eine Sammlung (Bibliothek) nützlicher allgemeiner Emacs Lisp-Funktionen namens **Elib** (auch von *Per Cederqvist* geschrieben, wird aber nicht zusammen mit `pcl-cvs`

verbreitet), um zu funktionieren.

Elib ist in der normalen Emacs-Distribution nicht enthalten (zumindest nicht im FSF Emacs; bei XEmacs weiß ich es nicht), sodass Sie Elib zunächst selber herunterladen und installieren müssen, bevor Sie `pcl-cvs` nutzen können. Sie können es von ftp://ftp.lysator.liu.se/pub/emacs/elib-1.0.tar.gz bekommen. Installationsanweisungen sind in dem Archiv enthalten.

Wenn Elib installiert ist, sind Sie in der Lage, `pcl-cvs` einzurichten. Die folgenden Anweisungen sind gleichermaßen für Version 1.05 und 2.x gültig (auch wenn Sie in den `NEWS`- und `INSTALL`-Dateien in neuen Distributionen nachschauen sollten, ob sich etwas geändert hat).

Entpacken Sie zunächst `pcl-cvs`. (Ich benutze Version 1.05, es könnte aber genauso gut 2.9.6 sein.)

```
user@linux ~/ # zcat pcl-cvs-1.05.tar.gz | tar xvf -

pcl-cvs-1.05/
pcl-cvs-1.05/README
pcl-cvs-1.05/NEWS
pcl-cvs-1.05/INSTALL
pcl-cvs-1.05/ChangeLog
pcl-cvs-1.05/pcl-cvs.el
pcl-cvs-1.05/pcl-cvs.texinfo
pcl-cvs-1.05/compile-all.el
pcl-cvs-1.05/pcl-cvs-lucid.el
pcl-cvs-1.05/pcl-cvs-startup.el
pcl-cvs-1.05/pcl-cvs.info
pcl-cvs-1.05/Makefile
pcl-cvs-1.05/texinfo.tex
```

und gehen Sie dann in das Hauptverzeichnis des Quelltextes:

```
user@linux ~/ # cd pcl-cvs-1.05/
```

In diesem Verzeichnis ist ein Makefile. Entsprechend der Anweisungen in der `INSTALL`-Datei müssen Sie nun einige Dateipfade am Anfang des Makefile anpassen und dann eingeben:

```
user@linux ~/ # make install
```

Wenn das funktioniert, toll! Allerdings gibt es manchmal auch eine Fehlermeldung. (Der `pcl-cvs`-Programmcode ist sehr portabel, die Installationsprozeduren sind es aber manchmal nicht!) Machen Sie Solgendes, wenn Sie eine Fehlermeldung erhalten:

```
user@linux ~/ # make clean
user@linux ~/ # make
```

Wenn bis dahin alles gut geht, dann haben diese beiden Kommandos einen beträchtlichen Teil der Installation bereits erledigt, indem sie alle nötigen Emacs Lisp-Dateien byte-kompiliert haben. (Byte-Compiling verwandelt eine Datei mit lesbarem Emacs Lisp-Programmcode - eine `.el`-Datei - in eine kompaktere und effizientere Repräsentation - eine `.elc`-Datei. Emacs kann `.elc`-Dateien effizienter laden und ausführen als `.el`-Dateien.)

Ich fahre nun fort, in der Annahme, dass das mit dem Byte-Compiling geklappt hat. Sollte das nicht der Fall

sein, keine Panik: Die `.elc`-Dateien sind ein Luxus, keine Notwendigkeit. Sie verbessern die Leistung ein wenig, aber `pcl-cvs` kann problemlos mit den normalen `.el`-Dateien laufen.

Falls das `make install` fehlgeschlagen ist, ist der nächste Schritt der, die Emacs Lisp-Dateien (`.elc` oder `.el`) in ein Verzeichnis zu packen, aus dem Emacs sie automatisch laden kann. Emacs hat ein spezielles Verzeichnis auf Ihrem System für solche lokal installierten Lisp-Dateien. Um dieses Verzeichnis zu finden - es enthält eine Datei namens `default.el` -, suchen Sie an den folgenden Orten in dieser Reihenfolge:

* `/usr/share/emacs/site/lisp/`
* `/usr/local/share/emacs/site-lisp/`
* `/usr/lib/emacs/site-lisp/`
* `/usr/local/lib/emacs/site-lisp/`

Wenn Sie Ihr `site-lisp`-Verzeichnis gefunden haben, kopieren Sie alle Lisp-Dateien dort hinein (dafür müssen Sie möglicherweise Superuser (root) sein):

```
user@linux ~/ # cp -f *.el *.elc /usr/share/emacs/site-lisp/
```

Im letzten Schritt muss nun Emacs mitgeteilt werden, wo sich die Eintrittspunkte zu den `pcl-cvs`-Funktionen befinden (wobei der Haupteintrittspunkt `cvs-update` ist), sodass Emacs weiß, wie es bei Bedarf den `pcl-cvs`-Code laden muss. Da Emacs grundsätzlich die Datei `default.el` liest, wenn er startet, müssen dort die Eintrittspunkte aufgelistet werden.

Glücklicherweise liegt bei `pcl-cvs` die für `default.el` notwendige Liste schon bei. Kopieren Sie also einfach den Inhalt von `pcl-cvs-startup.el` in die Datei `default.el` hinein (oder in Ihre `.emacs`-Datei, falls Sie die Installation nur für sich selbst vornehmen), und starten Sie Ihren Emacs neu.

Am besten kopieren Sie auch die `.info`-Dateien in Ihr Info-Verzeichnis, und fügen Sie `pcl-cvs` dem Inhaltsverzeichnis in der Datei dir hinzu.

4 Benutzung von pcl-cvs

Wenn es erst einmal installiert ist, ist `pcl-cvs` ganz einfach zu benutzen. Sie starten einfach die Funktion `cvs-update`, und `pcl-cvs` erzeugt eine Liste mit den Dateien, die in Ihrer Arbeitskopie verändert oder aktualisiert wurden. Von dieser Liste aus können Sie dann `Commits` durchführen, diffs machen und so weiter.

Da `cvs-update` der Haupteintrittspunkt ist, schlage ich vor, dass Sie ihn an eine einfach zu merkende Tastenkombination binden, bevor Sie weitermachen. Ich habe ihn in meiner `.emacs`-Datei an die Tastenkombination `Ctrl + c v` gebunden:

```
(global-set-key "\C-cv" 'cvs-update)
```

Sie können die Funktion aber auch einfach starten, indem Sie `M + x cvs-update` eingeben (auch bekannt als `Esc + x cvs update`.

Wenn es gestartet wurde, lässt `cvs-update cvs update` laufen, als ob es in dem Verzeichnis gestartet worden wäre, in dem sich die gerade in Emacs geladene Datei befindet - so, als hätten Sie dort auf der Kommandozeile `cvs update` eingegeben. Hier ist ein Beispiel für das, was Sie dann in Emacs sehen könnten:

Beispiel

```
PCL-CVS release 1.05 from CVS release &#36;Name: &#36;.
Copyright (C) 1992, 1993 Per Cederqvist
Pcl-cvs comes with absolutely no warranty; for details consult the manual.
This is free software, and you are welcome to redistribute it under certain
conditions; again, consult the TeXinfo manual for details.
Modified ci README.txt
Modified ci fish.c
---------- End ----
```

Zwei Dateien sind lokal verändert worden. (Manche Versionen von `pcl-cvs` zeigen die Unterverzeichnisse der Dateien mit an.) Der nächste logische Schritt ist nun, den `Commit` einer oder beider Dateien vorzunehmen, was durch die Buchstaben `ci` auf jeder der beiden Zeilen angezeigt wird. Für den `Commit` einer Datei bewegen Sie einfach den Cursor auf deren Zeile, und drücken `c`. Es öffnet sich dann ein Buffer, in den Sie eine Log-Mitteilung beliebiger Länge eingeben können (echtes Bearbeiten von Log-Mitteilungen ist der Hauptvorteil von `pcl-cvs` gegenüber der Kommandozeile). Wenn die Mitteilung geschrieben ist, drücken Sie `Ctrl + c Ctrl + c`, um den `Commit` durchzuführen.

Wenn sich der `Commit` auf mehrere Dateien, die sich eine Log-Mitteilung teilen, auswirken soll, dann markieren Sie zunächst die gewünschten Dateien mit der Taste `m`. Ein Sternchen erscheint neben jeder markierten Datei:

Beispiel

```
PCL-CVS release 1.05 from CVS release &#36;Name: &#36;.
Copyright (C) 1992, 1993 Per Cederqvist
Pcl-cvs comes with absolutely no warranty; for details consult the manual.
This is free software, and you are welcome to redistribute it under certain
conditions; again, consult the TeXinfo manual for details.
* Modified ci README.txt
* Modified ci fish.c
---------- End ----
```

Wenn Sie nun irgendwo `c` eingeben, dann sind davon alle (und nur die) markierten Dateien betroffen. Schreiben Sie die Log-Mitteilung, und übermitteln Sie die Daten wie gehabt mit `Ctrl + c Ctrl + c`.

Sie können auch `d` drücken, um `cvs diff` mit einer Datei (oder mehreren markierten Dateien) zu starten, und `f`, um eine Datei in Emacs zu bearbeiten. Es sind noch andere Kommandos möglich; drücken Sie `Ctrl + h m` im Buffer mit der Dateiliste, um zu sehen, was sonst noch geht.

5 Fehlerbehandlung in pcl-cvs

Das Programm `pcl-cvs` hatte schon immer eine etwas ungeschickte Art, mit Fehlermeldungen und Informationen von CVS umzugehen (auch wenn das möglicherweise in aktuelleren Versionen verbessert wurde). Wenn es auf eine Nachricht von CVS stößt, die es nicht kennt, wird es hysterisch, und Sie landen in einem Mail-Buffer, vorbereitet, um einen Bug-Report an den Autor von `pcl-cvs` abzuschicken. Dummerweise sind aber unter den CVS-Nachrichten, die `pcl-cvs` möglicherweise nicht erkennt, auch solche, die bei einer

problematischen Integration (Merge) von Quelltext entstehen, was zwar selten, aber doch immer mal wieder vorkommt.

Falls `pcl-cvs` Sie also plötzlich in einen Mail-Buffer umschaltet - keine Panik. Lesen Sie den vorbereiteten Text der E-Mail sorgfältig durch - die problematische CVS-Ausgabe sollte da irgendwo drin stehen. Wenn es aussieht wie ein fehlgeschlagenes merge, dann verwerfen Sie die E-Mail einfach, und starten Sie erneut `cvs-update`. Diesmal sollte es keine Probleme mehr geben, da die Integration des problematischen Quelltexts bereits stattgefunden hat.

6 Die Zukunft von pcl-cvs

Auch wenn ich den Eindruck erweckt haben mag, dass `pcl-cvs` kaum weiterentwickelt wird und eine riskante Investition sein könnte, scheint mir die Instabilität eine zeitweilige Erscheinung zu sein. *Stefan Monnier* reagiert sehr schnell auf Anfragen. (Ich schrieb ihn mehrfach an, während ich dieses Kapitel verfaßte, und erhielt immer sofort eine Antwort; er kümmert sich bereits um mehrere Fehler in der Version 2.9.6.) Sehr wahrscheinlich werden Sie zu dem Zeitpunkt, zu dem dieses Buch erschienen ist, bereits eine stabile Version 2.9.7 oder höher herunterladen können. [Anm.d.Übers.: 2.9.8 ist da.]

Ich habe sogar eine ermutigende E-Mail zu diesem Thema von *Greg Woods* erhalten, einem früheren Betreuer von `pcl-cvs`, die ich hier wiedergeben möchte:

E-Mail

```
From: woods@most.weird.com (Greg A. Woods)
Subject: Re: Status der pcl-cvs Betreuung, Stabilität der neueren Versionen?
To: kfogel@read-bean.com
Date: Sun, 29 Aug. 1999 18:59:19 -0400 (EDT)
[...]

Ich habe Stefans Versionen inzwischen einige Zeit benutzt und
tatsächlich meinen eigenen Entwicklungszweig aufgegeben.

Er hat eine Menge wirklich gute Arbeit in PCL-CVS gesteckt, und
abgesehen von einigen wenigen Merkwürdigkeiten in Version 2.9.6, die
ich inzwischen täglich nutze, ist es wirklich benutzbar (und es ist so
unendlich viel benutzbarer mit aktuellen CVS-Versionen als die
Version, die in der CVS-Distribution enthalten war! ;-).

Ich habe eine Datei pcl-cvs.README auf meinem ftp-Server abgelegt, in
der steht, dass die Dateien wirklich ziemlich alt sind (zumindest nach
Internet-Zeit ;-), und einen Verweis auf Stefans ftp-Server angegeben.
[...]
```

In einer späteren E-Mail sagte *Greg*, dass die FSF erwägt, `pcl-cvs` in ihre nächsten Ausgabe von Emacs (20.5) als Bestandteil zu integrieren, was den größten Teil der oben stehenden Installationshilfen überflüssig machen würde. Seufz. Manchmal ist es wirklich schwierig, mit der Entwicklung freier Software Schritt zu halten.

7 cvsutils: allgemeine Werkzeuge zur Nutzung mit CVS

Benötigt: Perl

URLs:

http://www.red-bean.com/cvsutils/ http://www.red-bean.com/cvsutils/releases/

Autoren: *Tom Tromey* (ursprünglicher Autor) und *Pavel Roskin* (momentaner Betreuer)

Die Sammlung kleiner Programme namens `cvsutils` arbeitet meist (aber nicht immer) ohne aktive Verbindung in der CVS-Arbeitskopie. Arbeitsgänge ohne aktive Verbindung werden ohne Kontakt zum Archiv durchgeführt, belassen die Arbeitskopie jedoch in einem Zustand, der jederzeit einen problemlosen späteren Kontakt zum Archiv gewährleistet. Dieses Verhalten kann sehr praktisch sein, wenn Ihre Netzwerkverbindung sehr langsam oder unzuverlässig ist.

Die `cvsutils`-Programme habe ich unten in der (meiner Meinung nach) ungefähren Reihenfolge ihrer Nützlichkeit aufgelistet, die nützlichsten zuerst. Zufällig entspricht diese Reihenfolge auch der Reihenfolge der Sicherheit der Programme. Sicherheit ist hier ein Thema, denn einige der Programme können durch ihren normalen Ablauf den Verlust lokaler Veränderungen oder Dateien in Ihrer Arbeitskopie bewirken. Lesen Sie daher zunächst sorgfältig die Beschreibungen durch, bevor Sie diese Werkzeuge benutzen.

Bemerkung:

> Diese Dokumentation ist auf dem Stand von Version 0.1.4. Lesen Sie sicherheitshalber die Datei README in neueren Versionen, um die aktuellsten Informationen zu erhalten.

`cvsu`

Gefahr: Keine
Kontakt zum Archiv: Nein

Dieses Werkzeug führt ein simuliertes `cvs update` ohne Kontakt zum Archiv durch, indem es die Zeitstempel aller Dateien mit den Aufzeichnungen in `CVS/Entries` vergleicht. So können Sie feststellen, welche Dateien lokal verändert wurden und welche nicht von CVS erfasst sind. Anders als `cvs update` aktualisiert `cvsu` Ihre Dateien nicht aus dem Archiv.

Obwohl es diverse Optionen versteht, wird `cvsu` meistens ohne jegliche Option gestartet:

```
user@linux ~/ # cvsu

? ./bar
? ./chapter-10.html
M ./chapter-10.sgml
D ./out
? ./safe.sh
D ./tools
```

Die Kürzel auf der linken Seite entsprechen denen von `cvs update`, außer dass `D` hier ein Verzeichnis symbolisiert. Dieses Beispiel zeigt leider nicht, wie `cvsu` sofort durchlief, wohingegen ein normales `cvs update` bei meiner lahmen Modemverbindung eine halbe Minute oder so gebraucht hätte.

Geben Sie

```
user@linux ~/ # cvsu --help
```

ein, um eine Liste möglicher Optionen zu erhalten.

cvsdo

Gefahr: Sehr gering
Kontakt zum Archiv: Nein

Dieser Befehl kann die Effekte von `cvs add` und `cvs remove` auf die Arbeitskopie simulieren, ohne dabei eine Verbindung zum Archiv aufzubauen. Natürlich müssten Sie dann noch immer die Änderungen per `commit` übermitteln, damit sie auch im Archiv gültig werden, aber wenigstens die Kommandos `add` und `remove` selbst können dadurch beschleunigt werden. Und so wird es benutzt:

```
user@linux ~/ # cvsdo add DATEINAME
```

oder

```
user@linux ~/ # cvsdo remove DATEINAME
```

Um eine Liste weiterer möglicher Optionen zu bekommen, benutzen Sie:

```
user@linux ~/ # cvsdo --help
```

cvschroot

Gefahr: Gering
Kontakt zum Archiv: Nein

Dieses Werkzeug kann die Arbeitskopie auf ein neues Archiv umstellen, falls das Archiv auf einen anderen Server umziehen sollte. Wenn dieser Fall eintritt, dann ändert sich keine der Revisionen, aber alle Arbeitskopien müssen ihre `CVS/Root`-(und möglicherweise `CVS/Repository`-)Dateien aktualisieren, damit sie den neuen Standort des Archivs beinhalten. Die Benutzung von `cvschroot` ist viel schneller als ein komplett neuer `Checkout`. Ein weiterer Vorteil ist, dass Sie lokale Änderungen nicht verlieren. Benutzung:

```
user@linux ~/ # cvschroot NEUES_ARCHIV
```

Zum Beispiel:

```
user@linux ~/ # cvschroot

:pserver:benutzer@neues.archiv.wo.auch.immer.com:/home/cvs/projekt
```

cvsrmadm

Gefahr: Mäßig

Kontakt zum Archiv: Nein

Entfernt alle `CVS/`-Verwaltungsverzeichnisse in Ihrer Arbeitskopie und hinterlässt einen Dateibaum, der dem gleicht, der durch `cvs export` erzeugt würde. Auch wenn Sie durch diesen Befehl keine lokalen Änderungen verlieren, wird Ihre Arbeitskopie anschließend keine CVS-Arbeitskopie mehr sein.

Mit Bedacht verwenden!

`cvspurge`

Gefahr: Mittel
Kontakt zum Archiv: Nein

Dieser Befehl löscht alle nicht von CVS kontrollierten Dateien aus Ihrer Arbeitskopie. Es nimmt keine lokalen Änderungen an von CVS kontrollierten Dateien zurück.

Mit Bedacht verwenden!

`cvsdiscard`

Gefahr: Recht beträchtlich
Kontakt zum Archiv: Unter Umständen

Dies ist das Gegenstück zu `cvspurge`. Anstatt unbekannte Dateien unter Beibehaltung lokaler Änderungen zu löschen, verwirft es alle lokalen Änderungen und behält unbekannte Dateien bei.

Mit extremer Vorsicht verwenden!

`cvsco`

Gefahr: Groß
Kontakt zum Archiv: Unter Umständen

Macht dasselbe wie `cvspurge` und `cvsdiscard` zusammen! Es löscht alle nicht von CVS kontrollierten Dateien aus Ihrer Arbeitskopie und verwirft alle lokalen Änderungen.

Nur mit wirklich paranoider Vorsicht anwenden!

`cvsdate`

Dieses Programm ist scheinbar unvollständig und wird möglicherweise niemals fertig gestellt. (Siehe die Datei README für Details.)

8 cvs2cl.pl: die Erzeugung von ChangeLogs im GNU-Stil aus CVS Logs

Benötigt: Perl

URL: URL: http://www.red-bean.com/~kfogel/cvs2cl.shtml

Das Programm `cvs2cl.pl` dient zur Verdichtung und Umformatierung der Ausgabe von `cvs log`, damit daraus eine `ChangeLog`-Datei im GNU-Stil erzeugt werden kann. ChangeLogs sind chronologisch geordnete Dokumente, welche die Veränderungen eines Projektes über die Zeit in einem Format beinhalten, das speziell auf leichte Lesbarkeit durch den Benutzer ausgerichtet ist (siehe untenstehende Beispiele).

Das Problem mit dem Befehl `cvs log` ist, dass er seine Ausgabe dateiweise organisiert, ohne zu berücksichtigen, dass dieselbe Log-Mitteilung, wenn sie fast im selben Moment in diversen Dateien auftaucht, impliziert, dass alle betroffenen Dateien Bestandteil eines einzigen `Commit` waren. Daher ist die Betrachtung der `cvs log`-Ausgabe in der Absicht, einen Überblick über ein Projekt zu erhalten, von vornherein zum Scheitern verurteilt - man kann nur den Entwicklungshergang einzelner Dateien zuverlässig verfolgen.

In dem `ChangeLog`, das `cvs2cl.pl` produziert, werden identische Log-Mitteilungen zusammengefasst, sodass ein `commit`, der mehrere Dateien umfaßt, auch nur als einzelner Eintrag auftaucht. Zum Beispiel:

```
user@linux ~/ # cvs3cl.pl

cvs log: logging .
cvs log: logging a-verzeichnis
cvs log: logging a-verzeichnis/unterverzeichnis
cvs log: logging b-verzeichnis

user@linux / # cat ChangeLog

...
1999-08-29 05:44 jrandom
*README (1.6), hello.c (2.1), a-verzeichnis/irgendwas.c (2.1),
a-verzeichnis/unterverzeichnis/fisch.c (2.1): Ich committe mit pcl-cvs
2.9, weil mir danach ist.
1999-08-23 22:48 jrandom
*README (1.5): [no log message]
1999-08-22 19:34 jrandom
*README (1.4): triviale Änderung
...

user@linux / #
```

Der erste Eintrag zeigt, dass der `Commit` von vier Dateien auf einmal mit der Log-Mitteilung: **Ich committe mit pcl-cvs 2.9, weil mir danach ist.** vorgenommen wurde. (Die Option `-r` wurde benutzt, um die Revisionsnummer jeder Datei mit dem Log-Eintrag anzuzeigen.)

Wie CVS selbst, nimmt auch `cvs2cl.pl` das aktuelle Verzeichnis als implizites Argument, kann aber auch mit bestimmten Dateien arbeiten, wenn diese auf der Kommandozeile angegeben werden. Es folgen einige der am häufigsten benutzten Optionen.

`-h, --help`

Zeigt eine kurze Hilfe zur Benutzung des Programms an.

`-r, --revisions`

Listet Revisionsnummern in der Ausgabe mit auf (s.o.). In Verbindung mit der Option `-b` werden Entwicklungszweige als `ZWEIGNAME.N` gelistet, wobei `N` die Revision auf dem Zweig ist.

`-t, --tags`

Listet symbolische Namen bei allen Revisionen, die welche besitzen, mit auf.

`-b, --branches`

Listet den Namen des Entwicklungszweiges für alle Revisionen auf diesem Zweig mit auf (siehe auch `-r`).

`-g OPTIONEN, --global-opts OPTIONEN`

Übergibt OPTIONEN als globale Argumente an cvs. Intern ruft `cvs2cl.pl` cvs auf, um die unformatierten Log-Daten zu erhalten; dabei werden OPTIONEN direkt hinter den cvs-Aufruf geschrieben. Um beispielsweise eine Beschränkung auf wichtige Meldungen und Kompression zu erreichen, kann man folgenden Aufruf verwenden:

```
user@linux ~/ # cvs2cl.pl -g "-Q -z3"
```

`-l OPTIONEN, --log-opts OPTIONEN`

Wie `-g`, nur dass die OPTIONEN hier als Befehlsoptionen, nicht global, übergeben werden. Um beispielsweise ein `ChangeLog` zu erzeugen, das nur `Commits` zwischen dem 26. Juli und dem 15. August anzeigt, könnten Sie folgenden Befehl verwenden:

```
user@linux ~/ # cvs2cl.pl -l "'-d1999-07-26<1999-08-15'"
```

Beachten Sie die doppelten Anführungszeichen - diese sind in Unix notwendig, da die Shell, die `cvs log` innerhalb von `cvs2cl.pl` aufruft, das <-Zeichen sonst als Zeichen zur Eingabeumleitung interpretieren würde. Daher müssen die Anführungszeichen als Teil des Arguments übergeben werden, was eine zweite **Schicht** von Anführungszeichen notwendig macht.

`-d, --distributed`

Schreibe ein eigenes `ChangeLog` in jedes Unterverzeichnis, das nur die `Commits` in jenem Verzeichnis beinhaltet. (Im Gegensatz zur Erzeugung eines einzigen `ChangeLogs`, welches das aktuelle Verzeichnis und alle Unterverzeichnisse desselben beinhaltet.)

9 cvslock: Sperren von Archiven

Benötigt: C-Compiler zur Installation; nichts Besonderes zur Laufzeit

URL: ftp://riemann.iam.uni-bonn.de/pub/users/roessler/cvslock/

Operationen auf dem Archiv (egal ob über cvs oder direkt an seinen Dateien ausgeführte) geschehen normalerweise nicht atomar. Das Programm `cvslock` sperrt ein CVS-Archiv (für Lese- oder Schreibzugriff) in derselben Art, wie CVS das auch macht, sodass CVS die Sperrung beachtet. Das kann beispielsweise nützlich sein, wenn Sie eine Kopie des gesamten Archivs machen möchten, dabei aber vermeiden wollen, dass Sie eventuell gerade laufende `Commit`-Vorgänge nur teilweise mitbekommen oder die `Lock`-Dateien anderer Benutzer mit kopieren.

Das `cvslock`-Archiv ist sehr gut eingerichtet und kann wie jedes GNU-Programm installiert werden. Hier ist der Mitschnitt eines Installationsvorgangs:

```
user@linux ~/ # zcat cvslock-0.1.tar.gz | tar xvf -

cvslock-0.1/
cvslock-0.1/Makefile.in
cvslock-0.1/README
cvslock-0.1/COPYING
cvslock-0.1/Makefile.am
cvslock-0.1/acconfig.h
cvslock-0.1/aclocal.m4
cvslock-0.1/config.h.in
cvslock-0.1/configure
cvslock-0.1/configure.in
cvslock-0.1/install-sh
cvslock-0.1/missing
cvslock-0.1/mkinstalldirs
cvslock-0.1/stamp-h.in
cvslock-0.1/cvslock.c
cvslock-0.1/cvslock.1
cvslock-0.1/snprintf.c
cvslock-0.1/cvslssh
cvslock-0.1/VERSION

user@linux / # cd cvslock-0.1
user@linux / # ./configure

...

user@linux / # make

gcc -DHAVE_CONFIG_H -I. -I. -I. -g -O2 -c cvslock.c
gcc -g -O2 -o cvslock cvslock.o

user@linux / # make install

...

user@linux / #
```

(Beachten Sie, dass Sie den Befehl `make install` möglicherweise als Superuser (**root**) ausführen müssen.)

Nun ist `cvslock` als `/usr/local/bin/cvslock` installiert. Wenn Sie es aufrufen, können Sie das Archiv mit `-d` oder über die `$CVSROOT`-Umgebungsvariable angeben, genau wie bei CVS selbst auch (das folgende Beispiel nutzt `-d`). Das einzige benötigte Argument ist der Name des Verzeichnisses, das gesperrt werden soll, relativ zum Hauptverzeichnis des Archivs. Dieses Verzeichnis und alle Unterverzeichnisse desselben werden dann gesperrt. In diesem Beispiel gibt es keine Unterverzeichnisse, daher wird nur eine Lock-Datei erzeugt:

```
user@linux ~/ # ls /usr/local/archiv/projekt/b-verzeichnis/

random.c,v

user@linux / # cvslock -d /usr/local/archiv projekt/b-verzeichnis
user@linux / # ls /usr/local/archiv/projekt/b-verzeichnis/

#cvs.rfl.cvslock.floss.27378 random.c,v

user@linux / # cvslock -u -p 27378 -d /usr/local/archiv
projekt/b-verzeichnis
user@linux / # ls /usr/local/archiv/projekt/b-verzeichnis/

random.c,v

user@linux / #
```

Beachten Sie, dass ich `-p 27378` angeben musste, als ich (mittels `-u` für `unlock`) die Sperre wieder aufhob. Das kommt daher, dass `cvslock` Unix-Prozessnummern zur Namensvergabe bei der Sperrung verwendet, um sicherzustellen, dass die Namen der Lock-Dateien einzigartig sind. Wenn Sie die Sperre nun entfernen, müssen Sie `cvslock` (mittels der Option `-p`) mitteilen, welche Instanz Sie aufheben möchten, auch wenn nur eine existiert. (Sie können jedoch `-p` auch ohne `-u` verwenden.)

Wenn Sie vorhaben, eine Weile im Archiv zu arbeiten und einiges direkt im Dateisystem zu ändern, dann können Sie die Option `-s` verwenden, um sich von `cvslock` eine neue Shell starten zu lassen. In diesem Fall benutzt es die `$SHELL`-Umgebungsvariable in Ihrer aktuellen Shell, um herauszufinden, welche Shell verwendet werden soll:

```
user@linux ~/ # cvslock -s -d /usr/local/archiv projekt
```

Die Sperre bleibt dann genau so lange bestehen, bis Sie die Shell verlassen, wonach sie automatisch aufgehoben wird. Sie können auch die `-c`-Option verwenden, um einen Befehl auszuführen, während das Archiv gesperrt ist. Genau wie bei `-s` werden zunächst die Lock-Dateien erzeugt, dann wird der Befehl ausgeführt und nach dessen Ende die Sperre wieder entfernt. Im folgenden Beispiel sperren wir das Archiv gerade lange genug, um eine Liste der Lock-Dateien anzeigen zu können:

```
user@linux ~/ # cvslock -c 'find . -name "*cvslock*" -print' -d
/usr/local/archiv projekt

cvslock: '/usr/local/archiv/projekt' locked successfully.
cvslock: starting'find . -name "*cvslock*" -print' ...
./a-verzeichnis/unterverzeichnis/#cvs.rfl.cvslock.floss.27452
./a-verzeichnis/#cvs.rfl.cvslock.floss.27452
./b-verzeichnis/#cvs.rfl.cvslock.floss.27452
./#cvs.rfl.cvslock.floss.27452
```

```
user@linux / # find /usr/local/archiv/projekt -name "*cvslock*" -print
user@linux / #
```

Das Kommando (das Argument zur Option `-c`) wird mit dem angegebenen Archivverzeichnis als Arbeitsverzeichnis gestartet.

Normalerweise erzeugt `cvslock` eine Lesesperre. Wenn gegen Schreibzugriff gesperrt werden soll, dann geht das mit der Option `-W`. (Sie können für die Lesesperrung auch `-R` angeben, aber das ist ja sowieso standardmäßig aktiv.) Entfernen Sie immer alle Sperren, wenn Sie mit Ihrer Arbeit fertig sind, sodass andere CVS-Prozesse nicht unnötig warten müssen.

Beachten Sie, dass `cvslock` auf der Maschine laufen muss, auf der sich das Archiv befindet - es gibt keine Möglichkeit, ein externes Archiv anzugeben. (Weitere Informationen erhalten Sie mit dem Befehl `man cvslock`, wodurch eine Anleitung angezeigt wird, die beim `make install` für `cvslock` mit installiert wurde.)

10 Weitere Pakete

Eine Vielzahl anderer Softwarepakete von Drittanbietern ist für CVS verfügbar. Es folgen Hinweise auf einige von ihnen.

CVSUp (Bestandteil des FreeBSD-Projektes)

`CVSUp` ist ein effizientes Werkzeug zum Spiegeln1 von Dateien im allgemeinen mit spezieller Unterstützung für das Spiegeln von CVS-Archiven. Das Betriebssystem *FreeBSD* benutzt dieses Programm, um Aktualisierungen seines Hauptarchivs an andere weiterzuverteilen, sodass die Benutzer bequem auf dem neuesten Stand bleiben können.

Für mehr und allgemeinere Informationen zu `CVSUp` schauen Sie auf http://www.polstra.com/projects/freeware/CVSup/ nach.

Informationen speziell zu *FreeBSD* gibt es unter http://www.freebsd.org/handbook/synching.html#CVSUP.

CVSWeb: Ein Web-Iterface für CVS-Archive

`CVSWeb` stellt ein Web-Interface zur Verfügung, mit dessen Hilfe man sich in CVS-Archiven umsehen kann. Ein passenderer Name wäre wohl `RCSWeb`, da es Ihnen genau genommen erlaubt, bestimmte Revisionen direkt anzusehen, mitsamt Log-Mitteilungen und `Diffs`. Obwohl ich selber es nie als besonders ansprechend empfunden habe, muss ich zugeben, dass es recht intuitiv ist, und eine Menge Server benutzen es.

Obschon die Software ursprünglich von *Bill Fenner* geschrieben wurde, steht die Version, die momentan am aktivsten weiterentwickelt wird, offensichtlich unter der Obhut von *Henner Zeller*, unter http://stud.fh-heilbronn.de/~zeller/cgi/cvsweb.cgi/.

Möglicherweise möchten Sie auch Fenners ursprüngliche Webseite unter http://www.freebsd.org/~fenner/cvsweb/ und die Aufbereitung der `CVSWeb`-Szene von *Cyclic Software* unter http://www.cyclic.com/cyclic-pages/web-cvsweb.html besuchen.

Das CVS-contrib/-Verzeichnis

Wie schon in Kapitel 4 erwähnt, wird eine Reihe von Werkzeugen von Drittanbietern zusammen mit CVS geliefert und ist dort im Verzeichnis `contrib/` zusammengefasst. Auch wenn mir keine formelle Regel bekannt ist, welche Werkzeuge mit CVS vertrieben werden, könnte dies ein laufender Versuch sein, die am häufigsten mit CVS verwendeten Werkzeuge zu finden und dort zu sammeln, sodass man stets weiß, wo man suchen muss. Bis es so weit ist, ist der beste Weg, um solche Programme zu finden, immer noch ein Blick auf die diversen CVS-Webseiten und das Fragen auf der Mailingliste.

11 Eigene Werkzeuge schreiben

CVS mag zuweilen wie ein wüster Haufen improvisierter Standards erscheinen. Da sind das RCS-Format, diverse Ausgabeformate (`history`, `annotate`, `log`, `update` usw.), diverse Formate für die Verwaltungsdateien der Archive und Arbeitskopien, das Client/Server-Protokoll, das Lock-Datei-Protokoll ... (Und? Ist es Ihnen schon langweilig geworden? Wissen Sie, Ich könnte fröhlich weitermachen.)

Erfreulicherweise bleiben diese Standards von Version zu Version recht konsistent - wenn Sie also ein Programm zur Nutzung mit CVS schreiben, dann brauchen Sie zumindest nicht zu befürchten, auf ein bewegliches Ziel schießen zu müssen. Zu jedem internen Standard gibt es üblicherweise eine Reihe von Leuten auf der *infocvs@gnu.org*-Mailingliste, die sich mit diesem sehr gut auskennen. (Einige von ihnen halfen mir weiter, während ich an diesem Buch schrieb.) Weiterhin ist da die Dokumentation (in Englisch), die mit der CVS-Distribution geliefert wird (insbesondere `doc/cvs.texinfo`, `doc/cvsclient.texi` und `doc/RCSFILES`). Zuletzt ist da immer noch der CVS-Quelltext selbst, das letzte Wort zu jeder Frage bezüglich Implementation oder Verhalten von CVS.

Mit all diesen Informationen zu Ihrer Verfügung gibt es keinen Grund zu zögern. Wenn Sie sich irgendein Programm vorstellen können, das Ihr Leben mit CVS einfacher machen könnte, dann los, schreiben Sie es - die Chancen stehen gut, dass andere Leute das auch schon immer haben wollten. Anders als eine Veränderung an CVS selbst kann ein kleines eigenständiges Hilfsprogramm sich sehr schnell weit verbreiten, was schnellere Rückmeldung für seinen Autor und zügigere Behebung von Fehlern für alle Benutzer bedeutet.

1. Anm. d. Übers: Spiegeln (mirroring): das Erstellen des Abbildes einer kompletten Verzeichnisstruktur an anderer Stelle