>> Gentoo Linux CVS Tutorial

[Bitte Kapitel auswählen]

1. Einleitung

1.1 Struktur des Tutorials

Dieses Tutorial besteht aus zwei Teilen. Im ersten Teil werden Sie lernen wie Sie CVS als Nicht-Entwickler benutzen, d.h. wie Sie Quelltexte eines Programms herunterladen und stets aktualisiert halten. Im zweiten Teil werden sie dann in die Benutzung von CVS als Entwickler eingeführt. Dabei werden Sie lernen wie Sie Dateien innerhalb von CVS ändern, hinzufügen oder löschen können. Wenn Sie CVS noch nicht verwendet haben, dann sollten Sie mit dem ersten Teil beginnen. Wenn Sie allerdings schon grundlegende Erfahrungen mit CVS haben, dann können Sie den ersten Teil dieses Tutorials überspringen, wobei Sie den ersten Teil vielleicht überfliegen sollten.

1.2 Was ist CVS und wofür brauche ich es?

CVS ist ein Client-Server-System, das es Entwicklern erlaubt ihre Projekte an einem zentralen Ort zu speichern, dem sogenannten **Repository**. Mit den CVS Client-Programmen können Entwickler nun den Inhalt des Repositorys ändern. Das CVS merkt sich jede Änderung am Inhalt der Dateien. Auf diese Weise wird eine komplette Geschichte des Projektes geführt. Entwickler können sich so ältere Versionen einer Datei oder eine Liste von Änderungen ansehen und andere nützliche Anwendungen ausführen.

1.3 Welche Rolle spielt CVS?

Viele Open-Source Projekte haben ihren eigenen CVS Server, die von den Entwicklern des Projekts als zentraler Ablageplatz für ihre gesamte Arbeit genutzt werden. Die Quelltexte werden häufig täglich geändert und verbessert. Dabei sind die Entwickler häufig auf der ganzen Welt verstreut, und doch können sie dank CVS zusammen an einem Projekt arbeiten ohne sich gegenseitig zu behindern.

1.4 CVS -- Die aktuellsten Entwickler-Quelltexte

Wenn die Entwickler mit ihren Änderungen zufrieden sind, dann packen sie ihr Projekt in eine .tar.gz-Datei und veröffentlichen es als neue offizielle Version. Doch aus verschiedenen Gründen sind diese Versionen manchmal nicht aktuell genug. Im ersten Teil dieses Tutorials wird Ihnen nun gezeigt, wie Sie CVS einsetzen, um den neuesten Stand der Entwicklung herunterzuladen und für sich selbst nutzen zu können.

1.5 Habe ich CVS installiert?

Bevor Sie CVS benutzen können, müssen Sie es auf Ihrem System installieren. Der einfachste Weg auszuprobieren, ob CVS schon installiert ist:

Befehlsauflistung 1

```
#cvs
```

Wenn ein CVS Befehl gefunden wird, dann ist CVS auf Ihrem System schon installiert. Sonst müssen Sie entweder ein Binär-Paket für Ihre Distribution suchen und installieren, oder CVS selber kompilieren und installieren.

Notiz
Anmerkung des Übersetzers: CVS ist natürlich auch im Portage Tree unter dev-util/cvs zu finden.

1.6 CVS selber kompilieren

Zunächst müssen Sie cvs-1.11.tar.gz von ftp://ftp.cvshome.org/pub.cvs-1.11/cvs-1.11.tar.gz herunterladen. Für den Fall, dass auf dem CVS ftp-Server eine neuere Version existiert, können Sie natürlich auch diese herunterladen. Dann führen Sie diese Befehle aus. (Die Ausgabe wird hier nicht angegeben):

Befehlsauflistung 2

```
# tar xzvf cvs-1.11.tar.gz
# cd cvs-1.11
# ./configure
# make
# make install
```

Nun sollte CVS auf Ihrem System installiert sein.

1.7 Die CVSROOT

Bevor wir beginnen, müssen Sie über einige Grundlagen Bescheid wissen. Um sich zu einem CVS Repository zu verbinden müssen Sie zunächst die CVSROOT (Das CVS Wurzelverzeichnis) kennen. Die CVSROOT ist eine Zeichenfolge, ähnlich einer URL, das dem cvs-Befehl mitteilt, wo sich das Repository befindet, und wie Sie sich mit diesem verbinden möchten. CVS benutzt verschiedene Formate für die CVSROOT-Zeichenfolge, je nachdem, ob es sich um ein lokales oder entferntes Repository handelt und welche Methode Sie verwenden, um sich mit diesem zu verbinden. Hier sind einige Beispiele für CVSROOTs mit der zugehörigen Erklärung...

1.8 Ein lokales Verzeichnis

Befehlsauflistung 3

```
CVSROOT=/home/cvsroot
```

Diese Einstellung würde man verwenden, wenn man ein lokales CVSROOT-Verzeichnis verwenden möchte. Dabei befindet sich das lokale CVSROOT im Verzeichnis /home/cvsroot. Natürlich wäre es auch möglich ein entferntes Verzeichnis mittels NFS an dieser Stelle gemounted zu haben.

1.9 Ein entfernter passwortgeschützter Server

Befehlsauflistung 4

```
CVSROOT=:pserver:cvs@foo.bar.com:/home/cvsroot
```

Dies ist ein Beispiel für eine CVSROOT auf dem entfernten Server foo.bar.com, wobei dort das Verzeichnis /home/cvsroot verwendet wird. Das vorangestellte ":pserver:" sagt dem client, dass Sie sich über das CVS eigene Passwort-Server-Protokoll anmelden wollen. Diese Methode wird häufig verwendet, um anonymen Benutzern den Zugang zu öffentlichen CVS-Servern zu gestatten.

1.10 Ein entfernter Zugang über rsh/ssh

Befehlsauflistung 5

```
CVSROOT=drobbins@foo.bar.com:/data/cvs
```

Dies ist ein Beispiel für einen Zugang über das RSH oder SSH Protokoll. Dabei wird der CVS Server versuchen, das Repository auf foo.bar.com mittels des Benutzerkontos drobbins zu erreichen. Wenn die CVS_RSH Umgebunsgvariable auf "ssh" gesetzt wurde, dann wird der CVS-Client die Verbindung über SSH aufbauen, sonst wird RSH verwendet. Der SSH-Zugang ist beliebt, wenn Sicherheit nötig ist. Allerdings geben weder RSH noch SSH anonymen Benutzern eine Möglichkeit sich einzuloggen. Um diese Zugriffsmethode verwenden zu können, brauchen sie ein Benutzerkonto auf dem System foo.bar.com.

1.11 Einige weitere Dinge...

Zusätzlich zur CVSROOT müssen Sie den Namen des Moduls (eine Sammlung von Quelltextdateien) wissen, den Sie "auschecken" möchten. Weiterhin brauchen Sie noch ein anonymes Passwort, um sich auf dem CVS-Passwort-Server einzuloggen. Es gibt hier kein Standardformat, wie etwa beim anonymen FTP, daher müssen Sie das anonyme Passwort auf der Entwickler-Webseite nachlesen. Wenn Sie all diese Informationen haben, können Sie loslegen.

1.12 Erste Schritte mit CVS, Teil 1

Um Quelltexte von einem CVS-Server herunterzuladen sind zwei Schritte notwendig. Zunächst loggen wir uns beim Passwort-Server ein. Dann laden wir die Quelltexte mit dem *checkout* Befehl herunter. Hier ein Beispiel von Befehlen, um die aktuellen Quelltexte des Samba Projektes herunterzuladen:

Befehlsauflistung 6

```
# export CVSROOT=:pserver:cvs@pserver.samba.org:/cvsroot
```

Dieser erste Befehl setzt die CVSROOT-Umgebungsvariable. Wenn Sie diese Variable nicht setzen, müssen Sie die folgenden zwei Befehle jeweils durch ein *-d:pserver:cvs@pserver.samba.org:/cvsroot* hinter dem *cvs* Befehl erweitern. Die CVSROOT-Umgebungsvariable spart uns also einiges an Tipparbeit.

1.13 Erste Schritte mit CVS, Teil 2

Nun besprechen wir die Befehle, die notwendig sind, um eine aktuelle Kopie der Quelltexte zu erhalten. Weiter unten werden wir näher auf die genaue Bedeutung dieser Befehle eingehen.

Befehlsauflistung 7

```
#cvs login
(Logging in to cvs@pserver.samba.org)
CVS password: (Geben Sie hier ihr Passwort ein)

#cvs -z5 co samba
U samba/COPYING
U samba/Manifesst
U samba/README
U samba/Read-Manifest-Now
U samba/Roadmap
U samba/WHATSNWE.txt
(Dies ist nur ein Ausschnitt aus der eigentlichen cvs co Ausgabe
```

1.14 Erste Schritte mit CVS - Die Erklärung

Der erste Befehl loggt uns auf dem pserver ein. Der zweite Befehl sagt dem CVS Client, dass er das Samba-Modul auschecken soll (co - check out). Dabei wird ein gzip-Kompressions-Level von 5 ("-z5") verwendet, so wird die Übertragung über einen langsamen Internetzugang beschleunigt. Für jede Datei die auf dem lokalen System neu erzeugt wird, gibt CVS eine "U [Pfad] "-Zeile aus. Das "U" steht dabei für "update".

1.15 Nach dem Checkout

Wenn der checkout-Befehl fertig ist, sehen Sie ein Verzeichnis "samba" in ihrem aktuellen Arbeitsverzeichnis. In allen Unterverzeichnissen werden Sie Unterverzeichnisse mit dem Namen "CVS" finden. In diesen speichert CVS Informationen über die Verzeichnisse; sie können ignoriert werden, da sie für die Benutzung von CVS nicht von Bedeutung sind. Von nun an, brauchen Sie sich keine Sorgen mehr um die CVSROOT-Umgebungsvariable machen. Sie brauchen auch den "-d"-Teil der Komamndozeile nicht mehr. Diese Informationen befinden sich nun alle in den CVS-Unterverzeichnissen. Also, Sie brauchen die CVSROOT-Variable wirklich nur für den ersten Login und das erste Checkout.

1.16 Quelltexte aktualisieren

Jetzt haben Sie die Quelltexte. Sie können diese kompilieren, installieren, einsehen oder was auch immer Sie damit anstellen möchten.

Hin und wieder wird des aber vorkommen, dass Sie diese Quelltexte auf den neusten Stand bringen möchten. Dazu müssen Sie sich nicht wieder am pserver einloggen. Ihre Informationen sind ja noch im CVS Unterverzeichnis gespeichert. Gehen sie also zunächst in das Hauptverzeichnis des Moduls, das Sie augecheckt haben, geben Sie dann ein:

Befehlsauflistung 8

```
# cvs update -dP
```

1.17 Ein genauerer Blick auf "cvs update", Teil 1

Wenn es neue Dateien gibt, gibt CVS für jede Datei "U [Pfad]" aus, während die Datei gespeichert wird. Außerdem werden Sie einige Meldungen der Art "? [Pfad]" sehen, wenn Sie das Modul kompiliert haben. Dabei handelt es sich um Objekt-Dateien. CVS erkennt, dass diese Dateien nicht im entfernten CVS-Repository gespeichert sind.

1.18 Ein genauerer Blick auf "cvs update", Teil 2

Beachten Sie auch die beiden Befehlszeilenparameter, die wir verwendet haben. "-d" teilt CVS mit, dass neue Verzeichnisse angelegt werden sollen, wenn Sie auf dem entfernten Server eingerichtet worden sind. Dies würde sonst nicht geschehen. "-P" veranlaßt CVS leere Verzeichnise, die auf dem Server gelöscht wurden, auch im lokalen System zu löschen. CVS neigt sonst dazu viele nicht mehr benutzte Verzeichnisse anzusammeln.

Wenn Sie nur die neuesten Quelltexte herunterladen möchten, ist das eigentlich schon alles was Sie benötigen. Nun werden wir noch einen Blick darauf werfen, was Sie benötigen, wenn Sie mit CVS als Entwickler umgehen müssen.

2. CVS für Entwickler

2.1 Dateien modifizieren

Als Entwickler werden Sie Dateien ändern wollen, die mittels CVS organisiert sind. Dazu müssen Sie nur wie gewohnt die lokale Datei auf ihrem System bearbeiten. Ihre Änderungen werden nicht auf dem Server geändert bis Sie CVS ausdrücklich den "commit"-Befehl (commit - einreichen) erteilen. Wenn Sie alle Ihre Änderungen sorgfältig getestet haben, und Sie sich sicher sind, dass alles ordentlich funktioniert, brauchen Sie nur diese beiden Schritte zu befolgen. Zunächst gehen Sie sicher, dass Sie die aktuellsten Quelltexte auf Ihrem lokalen System besitzen. Dazu geben Sie folgenden Befehl ein.

Befehlsauflistung 9

```
# cvs update -dP
```

2.2 CVS berücksichtigt Änderungen Anderer

Wie Sie bereits gesehen haben, bringt "cvs update" Ihre Quellen auf den neuesten Stand. Aber was passiert mit Ihren Änderungen? Keine Sorge, diese gehen nicht verloren. Wenn ein anderer Entwickler eine Änderung an einer Datei vorgenommen hat, deren lokale Version Sie nicht verändert haben, so wird CVS die Änderung einfach übernehmen.

Wenn Sie die Zeilen 1-10 einer Datei geändert haben, ein zweiter Entwickler gleichzeitig die Zeilen 30-40 geändert hat, und diese vor Ihnen ins CVS Repository eingereicht hat, so wird CVS diese Dateien intelligent in Ihre lokale Kopie einbinden. So verlieren Sie Ihre eigenen Änderungen nicht. Durch dieses sogenannte "merging" können zwei oder mehr Entwickler gleichzeitig an einer Datei arbeiten.

2.3 Merging ist nicht perfekt!

Haben nämlich zwei Entwickler gleichzeitig **die selbe Region** einer Datei geändert, so wird es etwas kompilizierter. CVS wird Sie dann über einen Konflikt informieren. Auch wenn natürlich keine Arbeit verloren geht, ist etwas manuelle Arbeit erforderlich. CVS braucht jetzt Ihre Eingabe um zu entscheiden wie die Änderungen zu "mergen" sind.

2.4 "commit" - Übergeben - Oder, wie man seine Arbeit einreicht

Nun werden wir einen genauen Blick darauf werfen, wie die oben erwähnten Konflikte ausgeräumt werden können. Zunächst werden wir allerdings annehmen, dass keine Konflikte vorliegen, als Sie "cvs update -dP" eingegeben haben. Wenn keine Konflikte durch Ihre lokalen Dateien auftreten und Ihre lokalen Dateien auf dem aktuellen Stand sind, sind Sie bereit Ihre Änderungen an das CVS zu übergeben.

Befehlsauflistung 10

```
# cvs commit
```

2.5 Was "commit" eigentlich macht

"cvs commit" reicht nicht **nur** Ihre Änderungen ein. Bevor die Änderungen eingereicht werden, wird Ihr Editor gestartet. Hier können Sie nun eine Beschreibung Ihrer Änderungen eingeben. Speichern Sie die Datei und verlassen Sie den Editor, werden Ihre Änderungen und die zugehörigen Kommentare den anderen Entwicklern zugänglich gemacht.

2.6 Die Log-Datei ansehen

Es ist sehr einfach sich die komplette Änderungsgeschichte einer Datei sowie die entsprechenden Kommentare anzeigen zu lassen. Um diese Informationen zu sehen, geben Sie folgendes ein:

Befehlsauflistung 11

```
# cvs log meineDatei.c
```

Der "cvs log" Befehl ist rekursiv. Möchten Sie also die gesamten Änderungen eines ganzen Verzeichnisses anzeigen lassen, so wechseln Sie in das entsprechende Verzeichnis, bevor Sie folgenden Befehl eingeben:

Befehlsauflistung 12

```
# cvs log | less
```

2.7 Optionen des "commit"-Befehls

Sie möchten vielleicht einen bestimmten Editor verwenden, um Ihre Log-Einträge zu machen. Dazu können Sie einfach die "EDITOR"-Umgebungsvariable ändern. Es ist eine gute Idee diese Einstellung mittels Ihrer ~/.bashrc-Datei permanent einzurichten.

Befehlsauflistung 13

```
export EDITOR=jpico
```

Alternativ zum Verwenden eines Editors können Sie CVS den Log-Eintrag auch direkt über die Kommandozeile mitteilen. Verwenden Sie hierzu die -m-Option.

Befehlsauflistung 14

```
# cvs commit -m 'Ein paar blöde Fehler beseitigt'
```

2.8 Die .cvsrc Datei

Bevor wir uns einige weitere CVS-Befehle ansehen, empfehle ich Ihnen sich eine ~/.cvsrc-Datei anzulegen. Mittels dieser Datei können Sie CVS einige standard Befehlszeilenparameter mitteilen, sodass Sie diese nicht jedes Mal eingeben müssen.

Befehlsauflistung 15

```
cvs -q
diff -u -b -B
checkout -P
update -d -P
```

2.9 Die .cvsrc Datei - Die Erklärung

Die erste Zeile der Datei bringt CVS in den "quiet"-(leise)-Modus. So wird die Ausgabe des "cvs update"-Befehls deutlich lesbarer. Außerdem können Sie sobald sie diese Datei eingerichtet haben, einfach "cvs update" anstatt "cvs update -dP" eingeben.

2.10 Eine neue Datei in das Repository einfügen

Es ist sehr einfach eine neue Datei in CVS einzubinden. Zunächst erstellen Sie eine neue Datei wie üblich. Dann geben Sie diesen Befehl ein:

Befehlsauflistung 16

```
# cvs add meineDatei.c
cvs server: use 'cvs commit' to add this file permanently
```

Die Datei wird beim nächsten Ausführen von "cvs commit" dem Repository hinzugefügt werden. Vorher werden andere Entwickler diese Datei nicht sehen!

2.11 Ein Verzeichnis in das Repository hinzufügen

Der Vorgang zum Hinzufügen eines Verzeichnises ist ähnlich:

Befehlsauflistung 17

```
# mkdir foo
# cvs add foo
Directory /home/cvsroot/meinCode/foo added to the repository
```

Anders als bei einer Datei, wird ein neues Verzeichnis sofort für andere Entwickler sichtbar. Es ist also kein "cvs commit" erforderlich. Sobald Sie eine lokale Datei in das neue Verzeichnis speichern, werden Sie sehen, dass ein neues Unterverzeichnis namens "CVS" eingerichtet wird. Dieses Verzeichnis dient als Aufbewahrungsort für die Daten, die notwendig sind, um die Dateien zu verwalten. So läßt sich leicht erkennen, ob sich ein bestimmtes Verzeichnis im CVS Repository befindet, indem man nach einem CVS-Unterverzeichnis sucht.

2.12 "cvs add" - Notizen

Wie Sie sich vielleicht gedacht haben, bevor eine Datei CVS hinzugefügt werden kann, müssen Sie sicherstellen, dass das Verzeichnis im CVS Repository ist. Sonst werden Sie diesen Fehler zu sehen bekommen:

Befehlsauflistung 18

```
# cvs add meineDatei.c
cvs add: cannot open CVS/Entries for reading: No such file or directory
cvs [add aborted]: no repository
```

2.13 Ein genauerer Blick auf "cvs update", Teil 1

Bevor wir uns genauer ansehen, wie Konflikte gelöst werden, sollten wir uns mit der Ausgabe des "cvs update"-Befehls vertraut machen. Wenn Sie eine ~/.cvsrc-Datei mit der Zeile "cvs -q" eingerichtet haben, wird die Ausgabe deutlich einfacher zu lesen sein. "cvs update" informiert Sie detailiert darüber was es gerade unternimmt, indem es einen Buchstaben ausgibt, gefolgt von einem Leerzeichen und dem Namen einer Datei. Zum Beispiel:

Befehlsauflistung 19

```
# cvs update -dP
? distfiles
? packages
? profiles
```

2.14 Ein genauerer Blick auf "cvs update", Teil 2

"cvs update" verwendet das Fragezeichen, um Ihnen mitzuteilen, dass es nichts über diese Datei weiß. Diese Dateien und Verzeichnisse befinden sich nur in dem lokalen Verzeichnis, und sind nicht Teil des offiziellen Repositories. Und sie wurden auch nicht mit dem "add"-Befehl hinzugefügt. Hier ist eine Liste mit weiteren Buchstaben und deren Bedeutung:

Befehlsauflistung 20

```
U [Pfad]
```

Wird ausgegeben, wenn eine neue Datei oder ein neues Verzeichnis in ihrem lokalen Verzeichnis erzeugt wird, oder wenn eine Datei, die Sie nicht geändert haben, aktualisiert (U wie Update) wird.

Befehlsauflistung 21

```
A [Pfad]
```

Diese Datei ist vorgesehen in das Repository hinzugefügt zu werden, und wird beim nächsten "cvs commit" aufgenommen.

2.15 Ein genauerer Blick auf "cvs update", Teil 3

Befehlsauflistung 22

```
R [Pfad]
```

Wie "A", sagt ein "R" aus, dass diese Datei für die Löschung vorgesehen ist. Also wird die Datei gelöscht, wenn Sie das nächste Mal "cvs commit" ausführen.

Befehlsauflistung 23

```
M [Pfad]
```

Das "M" bedeutet, dass diese Datei von Ihnen modifiziert worden ist. Außerdem kann es bedeuten, dass neue Änderungen vom Repository in die lokale Datei eingebunden wurden.

Befehlsauflistung 24

```
C [Pfad]
```

Der Buchstabe "C" bedeutet, dass ein Konflikt vorliegt. Manuelles einbinden der Änderungen wird nötig sein, bevor Sie "cvs commit" ausführen können.

2.16 Konflikte auflösen

Nun werden wir uns ansehen wie man einen Konflikt ausräumt. Ich bin sehr in das Gentoo Linux Projekt eingebunden, dort haben wir unsere eigenen CVS-Server unter cvs.gentoo.org. Wir Entwickler verbringen unsere Zeit hauptsächlich damit, die Quellen zu hacken, die im "gentoo-x86"-Modul abgelegt sind. In diesem Modul haben wir eine Datei namens "ChangeLog", die (wie Sie bereits erraten haben) eine Beschreibung der wichtigsten Änderungen beinhaltet.

2.17 Ein Beispiel-Konflikt

Da diese Datei fast jedes Mal geändert wird, wenn ein Entwickler eine Änderung vornimmt, ist sie eine der Hauptquellen für Konflikte. Nehmen wir uns diesen Beispielkonflikt vor: Nehmen wir an, dass ich die folgenden Zeilen hinzugefügt habe.

Befehlsauflistung 25

```
date 25 Feb 2001

This is the thing I added myself
```

Nehmen wir ferner an, dass bevor ich die Möglichkeit hatte "cvs commit" durchzuführen, jemand anders diese Änderung vorgenommen hat.

Befehlsauflistung 26

```
date 25 Feb 2001

This is the part added by another developer
```

2.18 Ein Beispielkonflikt - Teil 2

Wenn ich nun "cvs update -dP" ausführe (was man immer tun sollte, vor einem "commit"), so erhalte ich eine Konfliktmeldung, da CVS nicht in der Lage ist meine lokale Änderung selbstständig in das CVS Repository einzufügen.

Befehlsauflistung 27

```
RCS file: /home/cvsroot/gentoo-x86/ChangeLog,v
retrieving revision 1.362
retrieving revision 1.363
Merging differences between 1.362 and 1.363 into ChangeLog
rcsmerge: warning: conflicts during merge
cvs server: conflicts found in ChangeLog
C ChangeLog
```

2.19 Konfliktlösung

Argh, ein Konlikt! Zum Glück ist die Lösung einfach. Ich starte meinen Texteditor, und sehe den Text am Anfang der Datei ChangeLog.

Befehlsauflistung 28

```
<<<<<<< ChangeLog
date 25 Feb 2001

This is the thing I added myself

=======
date 25 Feb 2001

This is the part added by another developer

>>>>>>> 1.363
```

2.20 Konfliktlösung, Teil 2

Anstatt eine der beiden Versionen zu bevorzugen, hat CVS einfach beide Versionen zusammengefügt, und mittels eindeutiger Zeichen erkenntlich gemacht, wo der Konflikt liegt. Nun ist es an mir, die entsprechenden Regionen durch den Text zu ersetzen, der wirklich dort stehen sollte. In diesem Fall, ist der zu ersetzende Text weder der eine noch der andere, beide sollen hintereinander in der Datei stehen.

Befehlsauflistung 29

```
date 25 Feb 2001

This is the thing I added myself

This is the part added by another developer
```

Nun habe ich die Konfliktregion in der Datei mit dem entsprechenden Text ersetzt (und die "======" gelöscht). Jetzt kann ich die Änderung mittels "cvs commit" problemlos übertragen.

2.21 Einige Tips zur Konfliktlösung

Wann immer Sie eine Datei editieren in der Konflikte aufgetreten sind, gehen Sie die gesamte Datei nach den Konfliktzeichen durch, so dass Sie alle Konfliktregionen bearbeiten! Wenn Sie einen Konflikt übersehen, wird CVS Ihnen nicht erlauben, die Änderungen zu übertragen. Es ist daher offensichtlich sehr wichtig alle Konfliktzeichen ("=====") zu löschen. Falls Sie einen Fehler beim Editieren der Datei machen, und dann Ihre Änderung speichern, können Sie immernoch eine Originalversion Ihrer Datei finden. Diese trägt den Namen ".#Dateiname.Version".

2.22 Eine Datei löschen

Befehlsauflistung 30

```
# rm meine_alte_datei.c
# cvs remove meine_alte_datei.c
```

2.23 Eine Datei löschen, weitergeführt

Die Datei wird durch den Befehl fürs Löschen vorgesehen, wenn Sie das nächste Mal ein "cvs commit" durchführen. Sobald Sie ein "commit" durchgeführt haben, wird die Datei offiziell vom CVS Repository gelöscht sein. CVS wird die Datei jedoch nicht einfach "wegwerfen", sondern vielmehr alle Daten im Zusammenhang mit dieser Datei behalten. Sowohl Inhalt, als auch Geschichte der Änderungen. Dies ist wieder ein Beispiel, wie CVS Ihren wertvollen Quelltext beschützt.

"cvs remove" ist rekursiv. Das bedeutet, dass Sie mehrere Dateien gleichzeitig löschen können, und dann "cvs remove" von einem Übergeordneten Verzeichnis aus aufrufen können, mit keinem weiteren Argument. Wenn Sie dies tun, werden alle Dateien als "zu-löschen" gekennzeichnet, bis Sie das nächste Mal "cvs commit" aufrufen. Dann werden alle gelöscht.

2.24 Ein Verzeichnis löschen

Wenn Sie ein gesamtes Verzeichnis löschen möchten, empfehle ich diese Vorgehensweise. Zunächst löschen Sie alle Dateien und entfernen Sie sie auch aus dem CVS Repository mittels "cvs remove", wie oben beschrieben.

Befehlsauflistung 31

```
#  rm *.c
#  cvs remove
```

2.25 Ein Verzeichnis löschen, weitergeführt

Dann übergeben Sie Ihre Änderungen an das CVS Repository.

Befehlsauflistung 32

```
# cvs commit
```

Nun folgt der Trick. Folgen Sie diesen Schritten, um das Verzeichnis wirklich zu löschen:

Befehlsauflistung 33

```
# cd ..
# cvs remove mein_Verzeichnis
# rm -rf mein_Verzeichnis
```

Bemerken Sie, dass das Löschen des Verzeichnisses keinen weiteren Aufruf von "commit" benötigt. Verzeichnisse werden in Echtzeit dem Repository hinzugefügt beziehungsweise gelöscht.

2.26 Das wars!

Das war Ihre Einführung in CVS. Ich hoffe, dass diese Tutorial Ihnen hilfreich war. Natürlich hat CVS eine Menge mehr Funktionen als die hier vorgestellten, aber dank der umfassenden Resourcen zum Thema CVS können Sie sich nach Bedarf weiter informieren.

- http://www.cvshome.org ist die Heimat der CVS Entwicklung. Dort finden Sie auch eine Menge an Dokumentationen zum Thema CVS, wie etwa die offizielle CVS-Online-Dokumentation
- Die CVS VersionControl for Web Site Projects Seite beinhaltet gute Informationen über CVS als Tool für die Entwicklung von Webprojekten.
- Karl Fogel hat ein Buch namens Open Source Development with CVS geschrieben. Einige Kapitel des Buches sind kostenlos online abrufbar.
- cvsweb ist ein wirklich gutes CGI Skript, das ein Web-Interface für Ihr CVS Repository zur Verfügung stellt. Sehr gut fürs Browsen.

2.27 Über dieses Dokument

Die ursprüngliche Version dieses Artikels wurde zuerst bei IBM developerWorks veröffentlicht und ist Eigentum von Westtech Information Services. Dieses Dokument ist eine erneuerte Version des Original-Artikels und enthält verschiedene Verbesserungen durch das Gentoo Linux Documentation Team.