CVS-Referenz

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

Der folgende Text enthält das Kapitel 9 der deutschen Übersetzung des Buches "Open Source Development with CVS", welche unter der GNU Public License veröffentlicht wurden. Die Originalversion dieses Textes ist unter http://cvsbook.redbean.com/cvsbook.html erhältlich. Wir haben an dieser Stelle das Inhaltsverzeichnis den tatsächlich enthaltenen Kapiteln angepaßt.

Das SelfLinux-Team

Inhaltsverzeichnis

1 Organisation und Konventionen

Dieses Kapitel ist eine vollständige Referenz für CVS-Kommandos, Archivverwaltungsdateien, Schlüsselwortersetzung, Laufzeit-Kontrolldateien, Arbeitskopien und Umgebungsvariablen - alles, was in CVS Version 1.10.7 (genauer: Stand 20. August 1999) enthalten ist.

Die Kommandos sind der wichtigste Teil jeder CVS-Referenz, also beginnen wir damit:

2 Kommandos

Dieser Abschnitt ist alphabetisch geordnet, um es einfacher für Sie zu machen, einen bestimmten Befehl oder eine Option zu finden. Die folgenden Konventionen werden verwendet:

Argumente zu Kommandos und Optionen sind in Großbuchstaben geschrieben und kursiv in der Zusammenfassung, die jede Erklärung einleitet. Optionale Teile werden in eckigen Klammern [] geschrieben. Wenn aus einer Menge ausgewählt werden muss, werden die Möglichkeiten durch ein Pipe-Symbol getrennt: **x|y|z**. Der Plural oder Punktierung **...** deuten auf mögliche Mehrfachoptionen hin, die üblicherweise durch Leerzeichen getrennt sind. Zum Beispiel bedeutet DATEIEN eine oder mehrere Dateien. Der Eintrag [&MOD...] ist also ein Und-Zeichen, das direkt von einem Modulnamen gefolgt wird, dann ein Leerzeichen und dann evtl. noch ein Und-Zeichen mit Modulnamen und so weiter - kein Mal oder häufiger. (Hier wurde die Punktierung verwendet, weil ein Plural nicht eindeutig ausgesagt hätte, ob das Und-Zeichen nur beim ersten Modul notwendig ist oder bei allen.) Wenn ein Plural in Klammern steht, wie beispielsweise bei DATEI(EN), dann bedeutet das, dass hier zwar mehrere Argumente stehen können, üblicherweise aber nur eines verwendet wird. REV wird oft als Platzhalter für einen Revisionsbezeichner verwendet. Letzterer ist üblicherweise lediglich eine Revisionsnummer oder ein Bezeichner. In ganz seltenen Fällen ist es möglich, dass nur die eine oder die andere Variante verwendet werden kann, aber diese Fälle werden im Text besonders angemerkt.

3 Typische Eigenschaften von CVS-Kommandos

CVS-Kommandos setzen sich wie folgt zusammen:

```
cvs [GLOBALE_OPTIONEN] KOMMANDO [OPTIONEN] [DATEIEN]
```

Der zweite Satz von Optionen wird bisweilen **Kommandooptionen** genannt. Weil es aber so viele davon gibt, werden sie hier üblicherweise nur **Optionen** genannt, um etwas Platz zu sparen.

Viele Kommandos sind dazu gedacht, in einer Arbeitskopie angewendet zu werden, und können daher ohne Dateiargument ausgeführt werden. Diese Kommandos beziehen sich dann standardmäßig auf das derzeitige Verzeichnis und alle darin enthaltenen Verzeichnisse. Wenn ich mich also im Text auf die **Datei** oder **Dateien** beziehe, dann ist die Rede von den Dateien, auf denen CVS-Operationen ausführt. Je nachdem, wie Sie CVS aufgerufen haben, können diese Dateien also in der Kommandozeile aufgeführt sein oder auch nicht.

3.1 Datumsformate

Viele Optionen benötigen ein Datumsargument. CVS akzeptiert eine große Menge an unterschiedlichen Datumsformaten - zu viele, um sie hier aufzulisten. Im Zweifelsfall halten Sie sich an das ISO 8601-Format:

```
1999-08-23
```

Das bedeutet 23. August 1999 (genau genommen ist **23 August 1999** ebenfalls eine absolut gültige Datumsbezeichnung, solange Sie nicht vergessen, sie in Anführungszeichen zu setzen). Wenn Sie außerdem eine

Tageszeit angeben müssen, dann geht das so:

```
"1999-08-23 21:20:30 CET"
```

Sie können sogar übliche englische Wörter als Zeitangabe einsetzen, wie zum Beispiel `now` (jetzt), `yesterday` (gestern) oder sogar `12 days ago`(vor 12 Tagen). Im Allgemeinen können Sie gefahrlos mit Datumsformaten experimentieren; wenn CVS Ihre Datumsangabe überhaupt versteht, dann mit großer Wahrscheinlichkeit auch so, wie Sie es sich gedacht haben. Wenn nicht, dann wird der Befehl mit einer Fehlermeldung beendet.

4 Globale Optionen

Hier folgt eine Liste aller globalen Optionen für CVS

4.1 --allow-root=ARCHIV

Die alphabetisch erste globale Option, die praktisch nie auf der Kommandozeile verwendet wird. Sie wird mit dem `pserver`-Kommando zum Aufbau eines authentifizierten Zugriffes zu dem angegebenen Archiv verwendet. Dabei handelt es sich um das Hauptverzeichnis eines Archivs, (wie `/usr/local/cvsarchiv/`), nicht um ein Unterverzeichnis (wie `/usr/local/cvsarchiv/projekt`). Normalerweise ist die einzige Stelle, an der diese Option überhaupt vorkommt in der Datei `/etc/inetd.conf` (siehe Kapitel 4), genau wie das mit ihr verwendete `pserver` Kommando. Jedes Archiv, auf das mit `cvs pserver` auf einem bestimmten Rechner zugegriffen wird, benötigt eine entsprechende `--allow-root`-Option in `/etc/inetd.conf`. Dies ist eine Sicherheitsmaßnahme, die gewährleisten soll, dass niemand mittels eines `cvs pserver`-Kommandos Zugriff auf private Archive erlangen kann. (Siehe auch den Abschnitt **Password Authentication Server** im Cederqvist.)

4.2 -a

Authentifiziert jede Kommunikation mit dem Server. Diese Option hat keinen Effekt, solange Sie nicht mittels `GSSAPI`-Server (gserver) Verbindungen aufbauen. `GSSAPI`-Verbindungen sind in diesem Buch nicht behandelt, weil sie noch immer relativ selten anzutreffen sind (auch wenn sich das ändern könnte). Siehe auch die Abschnitte **Global Options** und **GSSAPI Authenticated** im *Cederqvist* für mehr Information zu diesem Thema.

4.3 -b (Überholt)

Diese Option wurde früher dazu verwendet anzugeben, in welchem Verzeichnis sich die `RCS`-Programmdateien befanden. CVS implementiert inzwischen jedoch intern die `RCS`-Funktionen, sodass diese Option nur noch aus Gründen der Kompatibilität beibehalten wird. Sie hat keinerlei Wirkung mehr.

4.4 -d ARCHIV

Diese Option gibt das Archiv an. ARCHIV kann entweder einfach ein absoluter Pfad oder ein komplexerer Ausdruck sein, der eine Verbindungsmethode, einen Benutzer- und Rechnernamen sowie einen Pfad enthält. In letzterem Fall sieht die generelle Ausdrucksform so aus:

```
:METHODE:BENUTZER@RECHNER:PFAD_ZUM_ARCHIV
```

Hier ein paar Beispiele zu den üblichen Verbindungsarten:

`:ext:jrandom@floss.red-bean.com:/usr/local/archiv`
- Baut eine Verbindung unter Verwendung von `rsh`, `ssh` oder eines anderen externen Programmes auf. Falls die `CVS_RSH`-Umgebungsvariable nicht gesetzt ist, ist die Standardeinstellung `rsh`, ansonsten wird der Wert dieser Variablen verwendet.

`:server:jrandom@floss.red-bean.com:/usr/local/archiv`
- Wie `:ext:`, nutzt aber die interne `rsh`-Variante von CVS. (Diese Option ist möglicherweise nicht auf allen Plattformen verfügbar.)

`:pserver:jrandom@floss.red-bean.com:/usr/local/archiv`
- Baut mit Hilfe des Passwortauthentifizierungs-Servers eine Verbindung auf (siehe **Der Passwortauthentifizierungs-Server** in Kapitel 4; siehe auch das `login`-Kommando).

`:kserver:jrandom@floss.red-bean.com:/usr/local/archiv`
- Baut eine Verbindung mit Hilfe der Kerberos-Authentifizierung auf.

`:gserver:jrandom@floss.red-bean.com:/usr/local/archiv`
- Baut eine Verbindung unter Verwendung der `GSSAPI`-Authentifizierung auf.

`:fork:jrandom@floss.red-bean.com:/usr/local/archiv`
- Baut eine Verbindung zu einem lokalen Archiv auf, benutzt aber das `Client/Server`-Protokoll, anstatt direkt auf die Dateien zuzugreifen. Dies kann nützlich sein, um das Verhalten einer CVS-Konfiguration im Netz auf dem eigenen Rechner zu testen und Fehler zu suchen.

`:local:jrandom@floss.red-bean.com:/usr/local/newrepos`
- Greift direkt auf ein lokales Archiv zu, so als wäre nur der absolute Pfad angegeben worden.

4.5 -e EDITOR

Startet EDITOR für Ihre `commit`-Mitteilung, falls diese nicht auf der Kommandozeile mit der `-m`-Option angegeben wurde. Normalerweise startet CVS den Editor abhängig vom Inhalt der Umgebungsvariablen CVSEDITOR, VISUAL oder EDITOR (in dieser Reihenfolge), falls keine Mitteilung mit `-m` angegeben wird. Falls alles schief geht, versucht CVS, den verbreiteten Unix-Editor **vi** zu starten. Wenn Sie sowohl `-e` also auch `-m` mit `commit` verwenden, wird das `-e` zu Gunsten der bei `-m` angegebenen Mitteilung ignoriert. (Daher ist es unproblematisch, `-e` in einer `.cvsrc`-Datei zu verwenden.)

4.6 -f

Diese globale Option unterbindet das Lesen der `.cvsrc`-Datei.

4.7 --help [KOMMANDO]

4.8 -H [KOMMANDO]

Diese beiden Optionen bewirken dasselbe. Sollte kein KOMMANDO angegeben sein, dann wird in einer kurzen Ausgabe die Bedienung der Grundfunktionen erläutert.

4.9 --help-options

Gibt eine Liste aller verfügbaren Optionen für CVS samt kurzer Erklärung aus.

4.10 --help-synonyms

Gibt eine Liste aller CVS-Kommandos und ihrer Kurzformen (`up` für `update` usw.) aus.

4.11 -l

Unterdrückt das Mitschneiden dieses Kommandos in die `CVSROOT/history`-Datei im Archiv. Das Kommando wird zwar ausgeführt, aber es wird kein Eintrag in der `History`-Datei erzeugt.

4.12 -n

Ändert keine Datei in der Arbeitskopie oder im Archiv. Mit anderen Worten, diese Option führt zu einer Art Testlauf, in dem CVS zwar fast alle Kommandos abarbeitet, aber nichts wirklich verändert. Das kann nützlich sein, wenn Sie sehen

wollen, was ein Kommando bewirkte, wenn Sie es tatsächlich anwenden würden. Ein übliches Szenario ist, wenn Sie sehen möchten, welche Dateien in Ihrem Arbeitsverzeichnis inzwischen verändert wurden, diese aber noch nicht aktualisieren möchten (was die Veränderungen aus dem Archiv in Ihre Dateien übertragen würde). Durch die Ausführung des Kommandos `cvs -n update` können Sie so eine Zusammenfassung der Veränderungen angezeigt bekommen, ohne Ihre Arbeitskopie der Daten anzutasten.

4.13 -q

Diese Option befiehlt CVS, etwas **leiser** zu arbeiten und unnötige Informationen nicht auszugeben. Was als notwendig angesehen wird, hängt hierbei von dem Kommando ab. Zum Beispiel werden bei der Aktualisierung die Nachrichten unterdrückt, die CVS normalerweise ausgibt, wenn es in ein neues Unterverzeichnis der Arbeitskopie eintritt - die einzeiligen Statusmeldungen über veränderte oder aktualisierte Dateien werden jedoch trotzdem angezeigt.

4.14 -Q

Wie `-q`, nur dass CVS hier lediglich die allernötigsten Informationen ausgibt. Kommandos, deren einziger Zweck es ist, Ausgaben zu erzeugen (wie z.B. `diff` und `annotate`), werden davon natürlich nicht beeinflusst. Hingegen geben Befehle, die unabhängig von irgendwelchen Ausgaben ihre Aufgabe erfüllen können (wie z.B. `update` und `commit`), nichts aus.

4.15 -r

Bewirkt, dass neue Arbeitsdateien ohne Schreibzugriff erzeugt werden (hat also dieselbe Wirkung wie die CVSREAD-Umgebungsvariable). Wenn Sie diese Option verwenden, werden bei Dateneinspielungen und -aktualisierungen die Dateien in Ihrer Arbeitskopie als nur lesbar markiert (sofern Ihr Betriebssystem diese Option unterstützt). Genau genommen habe ich keine Ahnung, warum je jemand diese Option verwenden sollte.

4.16 -s VARIABLE=WERT

Hiermit wird die interne CVS-Variable namens VARIABLE auf den Wert WERT gesetzt. Auf der Seite des Archivs können die `CVSROOT/*info`-Kontrolldateien solche Variablen in Werte übersetzen, die mit der `-s`-Option gesetzt wurden. Wenn zum Beispiel die Datei `CVSROOT/loginfo` eine Zeile wie diese enthält

```
myproj /usr/local/bin/foo.pl ${=FISCH}
```

und dann jemand ein `commit` von myproj wie folgt startet

```
user@linux ~/ # cvs -s FISCH=Karpfen commit -m "Köder-Bug behoben"
```

dann wird das Skript `foo.pl` mit **Karpfen** als Argument gestartet. Man beachte jedoch die schräge Syntax: Das Dollarzeichen, die geschweifte Klammer und das Gleichheitszeichen - fehlt eines der drei, dann funktioniert die Zuweisung nicht korrekt. Variablennamen dürfen lediglich Buchstaben, Zahlen und Unterstriche enthalten. Obwohl es nicht unbedingt erforderlich ist, dass alle Buchstaben groß geschrieben werden, scheinen sich doch die meisten Leute an diese Konvention zu halten. Das `-s`-Flag kann in einem Kommando beliebig oft hintereinander verwendet werden. Falls sich jedoch eines der Kontrollskripten auf Variablen bezieht, die nicht beim Aufruf gesetzt wurden, dann wird zwar das Skript ausgeführt, aber eine Warnung an den Benutzer ausgegeben und keiner einzigen Variablen ihr Wert zugewiesen. Wenn zum Beispiel in `loginfo` Folgendes steht

```
myproj /usr/local/bin/foo.pl ${=FISCH} ${=VOGEL}
```

```
```

aber derselbe Befehl wie vorhin ausgeführt wird

```
user@linux ~/ # cvs -s FISCH=Karpfen commit -m "Köder-Bug behoben"
```

dann bekommt der Benutzer, der den Befehl gestartet hat, eine Warnung wie

```
loginfo:31: no such user variable ${=BIRD}
```

und das Skript `foo.pl` wird ohne Argumente ausgeführt. Wenn das Kommando jedoch so aussieht:

```
user@linux ~/ # cvs -s FISCH=Karpfen -s VOGEL=Geier commit -m "Köder-Bug
behoben"
```

dann wird keine Warnung ausgegeben, und `foo.pl` wird wie erwartet mit den Argumenten Karpfen und Geier aufgerufen. In beiden Fällen würde der `Commit` selbst korrekt durchgeführt.

Bemerkung

> Obwohl all diese Beispiele `commit` verwenden, kann die Zuweisung von Variablen mit jedem CVS-Kommando geschehen, das eine `CVSROOT/` Kontrolldatei auslöst - darum ist `-s` eine globale Option.
>
> (Siehe auch den Abschnitt Archivverwaltungsdateien weiter hinten in diesem Kapitel zur Expansion von Variablen in Kontrolldateien.)

4.17 -T DIR

Speichert alle temporären Dateien in DIR anstatt da, wo CVS sie normalerweise ablegt. (Dies überschreibt im Besonderen den Wert der Umgebungsvariablen TMPDIR, falls sie existiert.) DIR sollte ein absoluter Pfad sein. Diese Option ist nützlich, falls Sie als Benutzer keinen Schreibzugriff auf die üblichen temporären Verzeichnisse besitzen.

4.18 -t

Verfolgt die Durchführung eines CVS-Kommandos. Dies veranlasst CVS genau alle Schritte anzuzeigen, die es durchführt, um einen Befehl auszuführen. Das kann sich besonders im Zusammenhang mit der Option `-n` als nützlich erweisen, damit Sie sich so im Detail die Effekte eines ungewöhnlichen Befehles im Voraus ansehen können, ohne ihn wirklich auszuführen. Die Option ist ebenfalls praktisch, um herauszufinden, warum ein Kommando nicht funktioniert.

4.19 -v --version

Veranlasst CVS, seine vollständige Version und Urheberrechtsangaben auszugeben.

4.20 -w

Bewirkt, dass neue Arbeitsdateien mit Schreibzugriff ausgestattet werden (überschreibt damit jeden Wert, den eventuell die Umgebungsvariable CVSREAD haben könnte). Weil normalerweise sowieso alle Dateien mit Schreibzugriff versehen werden, wird diese Option nur selten benutzt.

Werden beide Optionen, `-r` und `-w`, angegeben, so gilt die `-w`-Option.

4.21 -x

Verschlüsselt jede Kommunikation mit dem Server. Diese Option hat keine Wirkung, solange nicht eine Verbindung via GSSAPI zum Server (gserver) besteht. GSSAPI-Verbindungen werden in diesem Buch nicht behandelt, weil sie immer noch eher selten benutzt werden - obwohl sich das ändern könnte). (Siehe auch die Abschnitte **Global Options** und **GSSAPI Authenticated** im *Cederqvist* mit mehr Informationen zu dieser Thematik.)

4.22 -z GZIPLEVEL

Stellt die Stärke der Datenkompression für die Verbindungen zum Server ein. Das Argument GZIPLEVEL muss eine Zahl zwischen eins und neun sein. Eins steht für minimale Kompression (dafür schnell); neun hingegen hat nur auf sehr schnellen Rechnern oder sehr langsamen Verbindungen Sinn (komprimiert sehr stark, braucht aber viel Rechenzeit). Die meisten Benutzer dürften Werte zwischen drei und fünf praktikabel finden. Das Leerzeichen zwischen `-z` und seinem Argument ist optional.

5 Liste der Befehle

m Folgenden eine Liste aller CVS-Befehle

5.1 add [OPTIONS] FILES

Alternativen: ad, new
Erfordert: Arbeitskopie, Archiv
Ändert: Arbeitskopie

Fügt eine neue Datei einem existierenden Projekt hinzu. Obwohl das Archiv zur Bestätigung kontaktiert wird, wird die Datei dem Archiv nicht wirklich hinzugefügt, bis später `commit` ausgeführt wird. (Siehe auch `remove` und `import`.)

Optionen:

* `-k METHODE`
  * Spezifiziert, dass die Datei mit der angegebenen Methode zur Schlüsselwortersetzung abgespeichert werden soll. Zwischen `-k` und dem Argument ist kein Leerzeichen. (Für eine Liste gültiger Methoden und Beispiele siehe auch den Abschnitt **Schlüsselwortersetzung** weiter hinten in diesem Kapitel.)

* `-m MITTEILUNG`
  * Vermerkt MITTEILUNG als ersten Eintrag zur Entstehung bzw. Beschreibung der Datei. Diese unterscheidet sich von dem Log-Eintrag bei späteren Revisionen - jede Datei hat nur genau eine Beschreibung. Die Beschreibung ist optional.

Bemerkung:

> In Version 1.10.7 gibt es einen Fehler in CVS, der zum Verlust der Beschreibung führt, wenn eine Datei mittels Client/Server CVS hinzugefügt wird. Alle anderen Funktionen des `add`-Kommandos werden trotzdem einwandfrei ausgeführt, falls das ein Trost sein sollte.

5.2 admin [OPTIONEN] [DATEIEN]

Alternativen: adm, rcs
Erfordert: Arbeitskopie, Archiv
Ändert: Archiv

Dieses Kommando fungiert als Schnittstelle zu diversen administrativen Aufgaben - um genau zu sein Aufgaben, die sich auf einzelne RCS-Dateien innerhalb des Archivs beziehen, wie zum Beispiel die Veränderung der Schlüsselwortersetzungs-Methode oder die Veränderung einer Log-Mitteilung, nachdem diese bereits per `commit` gespeichert wurden.

Obwohl `admin` rekursiv arbeitet, sofern keine Dateiargumente übergeben wurden, ist es in den meisten Fällen sinnvoller, Dateien konkret zu spezifizieren. Es hat nur selten Sinn, einen `admin`-Befehl auf alle Dateien eines Projektes oder auch nur eines ganzen Verzeichnisses anzuwenden. Dementsprechend beziehen sich alle Erklärungen, die mit einer **Datei** zu tun haben, im Folgenden auf die im Befehl angegebene Datei, manchmal Dateien.

Bemerkung:

> Falls es auf dem Rechner mit dem Archiv eine Benutzergruppe namens **cvsadmin** gibt, so dürfen nur zu dieser Gruppe gehörige Benutzer das `admin`-Kommando ausführen. (Ausnahme: `cvs admin -k` ist für alle erlaubt.) So können Sie admin für alle Benutzer sperren, indem Sie diese Gruppe leer lassen.

Optionen:

* `-AALTE_DATEI (Überholt)`
  * Hängt die RCS-Zugriffsliste der Datei ALTE_DATEI an die Zugriffsliste der behandelten Datei an. CVS ignoriert RCS-Zugriffslisten, daher ist diese Option nutzlos.

* `-a BENUTZER1 [,BENUTZER2...] (Überholt)`
  * Hängt die BENUTZER in der durch Kommata separierten Liste der Zugriffsliste der behandelten Datei an. Wie `-A` ist auch diese Option wirkungslos.

* `-bREV`
  * Setzt die Revision des Standardverzeichnisbaumes der Datei (üblicherweise des Wurzelverzeichnisses) auf REV. Normalerweise werden Sie diese Option nicht benötigen, da Sie die Revisionen, die Sie benötigen, über bindende Markierungen erhalten, aber Sie könnten Sie verwenden, um zu einer Vertriebsversion zurückzukehren, falls Sie entsprechende Archivbereiche führen. Es sollte sich zwischen `-b` und seinem Argument kein Leerzeichen befinden.

* `-cKOMMENTAR_PREFIX (Überholt)`
  * Setzt das Kommentarzeichen der Datei auf KOMMENTAR_PREFIX. Dieses wird weder von CVS noch von aktuellen RCS-Versionen verwendet. Diese Option ist daher wirkungslos.

* `-eBENUTZER1 [,BENUTZER2...] (Überholt)`
  * Gegenstück zu `-a`: Entfernt Benutzernamen aus der Zugriffsliste. Ebenso wirkungslos wie `-a` und `-A`.

* `-i oder -I (Überholt)`
  * Diese beiden sind so alt und überholt, dass ich nicht einmal auf ihre frühere Bedeutung eingehe. Neugierige mögen den *Cederqvist* befragen.

* `-kMETHODE`
  * Setzt die Methode der Schlüsselwortersetzung für die Datei auf METHODE. Diese Option verhält sich wie die `-k`-Option des `add`-Befehls, nur dass man hier die Methode ändern kann, nachdem die Datei dem Archiv hinzugefügt wurde. (Siehe auch den Abschnitt zur Schlüsselwortersetzung weiter hinten in diesem Kapitel.) Zwischen `-k` und seinem Argument darf sich kein Leerzeichen befinden.

* `-L`
  * Setzt das Locking der Datei auf **strikt** (siehe `-l`).

* `-l[REV]`
  * Legt die Revision der Datei auf REV fest. Fehlt REV, so wird die neueste verfügbare Revision auf dem Hauptentwicklungszweig gehalten. Wenn REV ein Zweig des Archivs ist, so wird die aktuellste Revision in diesem Zweig des Archivs festgehalten (**Locking**). Der Zweck dieser Option ist, Ihnen eine Möglichkeit zu geben, **reservierte `Checkouts`** zu machen, bei denen nur ein Benutzer auf einmal die Datei bearbeiten kann. Ich bin nicht sicher, wie nützlich diese Option wirklich ist, aber wenn Sie sie ausprobieren möchten, sollten Sie das eventuell in Verbindung mit dem `rcslock.pl`-Skript aus dem `contrib/`-Verzeichnis der CVS Quelltextdistribution machen. Siehe auch die Kommentare in jener Datei für weitere Informationen. Unter anderem weisen diese Kommentare darauf hin, dass das Locking auf strikt gesetzt werden muss (siehe `-L`). Zwischen `-l` und dem Argument ist kein Leerzeichen.

* `-mREV:MITTEILUNG`
  * Ändert die Log-Mitteilung für die Revision REV auf MITTEILUNG. Sehr praktisch - neben `-k` ist dies die wohl meist genutzte Option zu `admin`. Es sind keine Leerzeichen vor dem Argument oder um den Doppelpunkt herum gestattet. Natürlich darf MITTEILUNG trotzdem Leerzeichen enthalten (dann aber bitte die MITTEILUNG in Anführungszeichen setzen, damit die Shell nicht durcheinander kommt!).

* `-NNAME[:[REV]]`
  * Wie `-n`, nur dass es das Überschreiben jeder existierenden Zuweisung des symbolischen Namens NAME erzwingt, anstatt mit einer Fehlermeldung auszusteigen.

* `-nNAME[:[REV]]`
  * Es handelt sich hier um eine allgemeine Schnittstelle zur Zuweisung, Umbenennung und zum Löschen von Markierungen. Soweit ich das beurteilen kann, gibt es keinen Grund, diese Option dem `tag`-Kommando und seinen vielfältigen Optionen vorzuziehen (`-d`, `-r`, `-b`, `-f` usw.). Ich empfehle daher die Benutzung des `tag`-Befehls. Der NAME und die Optionale REV können sich auf folgende Arten zusammensetzen:
  * Falls nur das NAME-Argument angegeben wurde, wird der symbolische Name (tag) namens NAME gelöscht. Falls NAME: angegeben wurde, aber kein REV, so wird NAME der aktuellsten Revision des Standardverzeichnisbaumes (üblicherweise der Hauptentwicklungslinie) zugewiesen. Falls NAME:REV gegeben ist, wird NAME der Revision zugeordnet. REV kann wiederum ein symbolischer Name sein, dann wird es zunächst in eine Revisionsnummer übersetzt (kann auch die Nummer einer abgezweigten Version sein). Falls REV die Nummer einer abgezweigten Version ist und ein Punkt folgt (**.**), dann wird NAME der aktuellsten Revision auf dieser abgezweigten Version zugewiesen. Ist REV nur $, dann wird NAME den Revisionsnummern zugeordnet, die in den Schlüsselwörtern der behandelten Datei(en) gefunden werden.
  * In allen Fällen, in denen ein Name zugewiesen wird, beendet sich CVS mit einer Fehlermeldung, sofern schon eine Marke namens NAME in der Datei existiert. (Ausnahme: siehe `-N`.) Es sind keine Leerzeichen zwischen `-n` und seinen Argumenten.

* `-oBEREICH`
  * Löscht die Revisionen, die durch BEREICH spezifiziert werden (auch bekannt als **Outdating**, daher `-o`). BEREICH kann wie folgt angegeben werden:
    * `REV1::REV2`
      - Vernichtet alle Revisionen zwischen REV1 und REV2, sodass in der Revisionshistorie REV2 direkt auf REV1 folgt. Nach dieser Aktion sind alle Versionen dazwischen nicht mehr existent, und es gibt einen Sprung in der Folge der Revisionen.
    * `::REV`
      - Löscht alle Revisionen vom Anfang dieser abgezweigten Version (die auch die Hauptentwicklungslinie sein kann) bis hin zu REV, natürlich nicht inklusive REV. Danach ist REV die erste Revision in diesem Bereich.
    * `REV::`
      - Schreddert alle Revisionen, die in dieser abgezweigten Version (die auch die Hauptentwicklungslinie sein kann) auf REV folgen. REV ist dann die letzte Revision in diesem Bereich.
    * `REV`
      - Löscht Revision REV. (`-o1.8` wäre äquivalent zu `-o1.7::1.9.`)
    * `REV1:REV2`
      - Löscht von REV1 bis REV2 inklusive! Sie müssen im selben Entwicklungszweig sein. Danach ist es unmöglich, auf REV1, REV2 und alle Revisionen dazwischen zuzugreifen.
    * `:REV`
      - Löscht Revisionen vom Anfang des Entwicklungszweiges bis REV inklusive. (Siehe vorhergehende Warnung.)
    * `REV:`
      - Löscht beginnend mit REV bis zum Ende der abgezweigten Version inklusive. (Siehe vorhergehende Warnung.)

Bemerkung:

Keine der zu löschenden Revisionen darf Entwicklungszweige oder Locks haben. Sollten irgendwelchen der betroffenen Revisionen symbolische Namen zugewiesen sein, müssen diese zunächst mit `tag -d` oder `admin -n` gelöscht werden. (Genau genommen schützt CVS momentan nur dann gegen das Löschen symbolisch benannter Revisionen, wenn die **::**-Notation verwendet wird, aber die **:**-Notation könnte sich diesem Verhalten bald anpassen.)

Anstatt diese Option zu nutzen, um ein fehlerhaftes `Commit` zu löschen, sollten Sie eine neue Version per `Commit` in das Archiv bringen, welche die fehlerhafte Änderung wieder in Ordnung bringt. Es gibt keine Leerzeichen zwischen `-o` und seinen Argumenten.

* `-q`
  * Stellt CVS ruhig - keine unnötigen Mitteilungen werden generiert (genau wie mit der globalen Option `-q`).

* `-sSTATUS[:REV]`
  * Setzt das Statusattribut der Revision REV auf STATUS. Wird REV weggelassen, so wird die aktuellste Revision des Hauptentwicklungszweiges angenommen. Ist REV eine Markierung oder Nummer einer abgezweigten Version, dann wird die aktuellste Revision auf dieser abgezweigten Version verwendet. Beliebige Buchstaben- oder Nummernfolgen für STATUS werden akzeptiert. Einige übliche Statusbezeichnungen sind Exp für experimentell, Stab für stabil und Rel für released (veröffentlicht). (Tatsächlich setzt CVS den Status automatisch auf Exp, wenn eine Datei neu erzeugt wird.) Beachten Sie, dass CVS die Statusbezeichnung dead für eigene Zwecke verwendet, also geben Sie diese bitte nicht an. Statusinformationen werden in der `cvs log`-Ausgabe sowie in den `$Log-` und `$State-RCS`-Schlüsselwörtern in Dateien ausgegeben. Zwischen `-s` und seinem Argument sind keine Leerzeichen.

* `-t[BESCHREIBUNGSDATEI]`
  * Ersetzt die Beschreibung (den `Log`-Eintrag zur Entstehung) für die Datei durch den Inhalt von BESCHREIBUNGSDATEI oder liest diesen von der Standardeingabe, falls keine BESCHREIBUNGSDATEI angegeben wurde. Diese nützliche Option funktioniert - leider - momentan nicht im Client/Server-CVS. Außerdem wird jede existierende Beschreibung einer Datei gelöscht und durch einen leeren String ersetzt, falls man versucht, diese Option im Client/Server-CVS ohne Angabe von BESCHREIBUNGSDATEI zu verwenden. Wenn Sie die Beschreibung einer Datei erneuern müssen, dann tun Sie dies bitte nur lokal auf der Maschine mit dem Archiv, oder benutzen Sie die Option `-t`-BESCHREIBUNG. BESCHREIBUNGSDATEI darf nicht mit einem Bindestrich (-) beginnen (siehe `-t`-BESCHREIBUNG).

* `-t-BESCHREIBUNG`
  * Wie `-t`, nur dass hier BESCHREIBUNG direkt als Beschreibung verwendet wird. BESCHREIBUNG darf Leerzeichen enthalten, wenn sie mit Anführungszeichen eingeschlossen wird. Diese Variante funktioniert im Client/Server-CVS und auch lokal ohne Probleme.

* `-U`
  * Setzt das Locking auf **nicht strikt**. (Siehe auch `-l` und `-L` weiter oben.)

* `-u[REV]`
  * Hebt die Sperrung der Revision REV auf (siehe `-l`.) Wird REV weggelassen, so entsperrt CVS die aktuellste durch den Aufrufer gesperrte Revision. Ist REV ein Entwicklungszweig, dann entsperrt CVS die aktuellste gesperrte Revision in dieser abgezweigten Version. Wenn jemand anders als der Sperrer einer Revision die Sperre wieder aufhebt, so wird an denjenigen, der die Sperrung vorgenommen hat, eine E-Mail verschickt. Die

Person, welche die Sperre aufhebt, wird auf der Kommandozeile um eine Stellungnahme für die E-Mail gebeten. Zwischen `-u` und seinem Argument ist kein Leerzeichen.

* `-VRCS_VERSIONSNUMMER (Überholt)`
  * Dies war einmal ein Weg, um CVS zur Erzeugung von RCS-Dateien zu bewegen, die zu früheren RCS-Versionen kompatibel waren. Inzwischen weicht das RCS-Format, das CVS benutzt, zunehmend von dem RCS-Format, das RCS verwendet, ab, sodass diese Option nutzlos geworden ist. Die Angabe dieser Option führt zu einer Fehlermeldung.

* `-xENDUNG (Überholt)`
  * Theoretisch kann man mittels dieser Option die Dateiendung der RCS-Dateien bestimmen. Leider verlassen sich CVS und seine Werkzeuge darauf, dass diese Endung standardmäßig ist **(,v)**, daher bewirkt diese Option nichts.

5.3 annotate [OPTIONEN] [DATEIEN] (Überholt)

Alternativen: ann
Erfordert: Arbeitskopie, Archiv
Ändert: Nichts

Zeigt Informationen darüber an, wer zuletzt welche Zeile jeder Datei geändert hat und wann. Jede Ausgabezeile entspricht einer Zeile der Datei. Von links nach rechts beinhaltet die Zeile die Revisionsnummer der letzten Modifikation der Zeile, einen in Klammern stehenden Ausdruck, der den Benutzer und das Datum der letzten Änderung beinhaltet, einen Doppelpunkt und den Inhalt der Zeile in der Datei.

Wenn zum Beispiel eine Datei so aussieht:

Dies ist eine Testdatei

Sie hat nur zwai Zeilen

Ich meine zwei

dann können die Anmerkungen (`annotations`) für diese Datei so aussehen:

```
1.1 (jrandom 22-Aug-99): Dies ist eine Testdatei
1.1 (jrandom 22-Aug-99): Sie hat nur zwai Zeilen
1.2 (jrandom 22-Aug-99): Ich meine zwei
```

sodass Sie nun sehen, dass die ersten zwei Zeilen in der ersten Revision waren und die letzte Zeile in Revision 1.2 hinzugefügt oder geändert wurde (auch von jrandom).

Optionen:

* `-D DATUM`
  * Zeigt die Anmerkungen der letzten Revision vor DATUM.

* `-f`
  * Erzwingt die Anzeige der ersten Revision, falls die angegebene Markierung oder ein angegebenes Datum nicht gefunden wurde. Sie können diese Option in Verbindung mit `-D` und `-r` verwenden, um sicherzustellen, dass in

jedem Fall eine Ausgabe erfolgt, auch wenn es nur Revision 1.1 der Datei ist.

* `-l`
  * Lokal. Der Befehl bezieht sich nur auf das aktuelle Arbeitsverzeichnis. Eventuelle Unterverzeichnisse werden nicht behandelt.

* `-R`
  * Rekursiv. Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

* `-r REV`
  * Zeigt nur Anmerkungen zur Revision REV (kann eine Revisionsnummer oder eine Marke sein).

5.4 checkout [OPTIONEN] PROJEKT(E)

Alternativen: co, get
Erfordert: Archiv
Ändert: Aktuelles Verzeichnis

Extrahiert ein Modul aus dem Archiv in eine lokale Arbeitskopie. Die Arbeitskopie wird neu erzeugt, falls sie nicht existiert, und aktualisiert, falls sie bereits existiert (siehe auch `update`.)

Optionen:

* `-A`
  * Setzt alle bindenden Markierungen (Sticky Tags), Datumsangaben oder Schlüsselwortersetzungs-Modi zurück. Diese Option entspricht der `-A`-Option für den `update`-Befehl und wird dort wahrscheinlich häufiger genutzt als hier bei `checkout`.

* `-c`
  * Bewirkt, dass praktisch kein `Checkout` stattfindet; nur die `CVSROOT/modules`-Datei wird sortiert auf die Standardausgabe ausgegeben. Dies ist eine gute Methode, um einen Überblick darüber zu bekommen, welche Projekte sich in einem Archiv befinden. Projekte ohne Eintrag in der modules-Datei werden jedoch nicht aufgelistet. (Das ist ganz normal, weil der Name des Hauptverzeichnisses eines Projektes im Archiv standardmäßig als der Modulname des Projektes dient.)

* `-D DATUM`
  * Führt einen `Checkout` der letzten Revisionen bis DATUM aus. Diese Option ist bindend, d.h. Sie können anschließend aus der Arbeitskopie keinen `Commit` mehr durchführen, ohne das gebundene Datum zurückzusetzen (siehe `-A`). Diese Option impliziert `-P`, siehe unten.

* `-d VERZ`
  * Erzeugt die Arbeitskopie in einem Verzeichnis VERZ, anstatt ein Verzeichnis mit dem Namen des vom `Checkout` betroffenen Moduls zu erzeugen. Wenn Sie nur einen Teil eines Projektes behandeln und dieser Teil irgendwo unterhalb des Wurzelverzeichnisses des Projektes liegt, werden die lokalen leeren Zwischenverzeichnisse weggelassen. Sie können `-N` benutzen, um diese Sparmaßnahme zu unterbinden.

* `-f`

    * Erzwingt einen `Checkout` der ersten Revision, falls die angegebene Marke oder das angegebene Datum nicht gefunden wurde. Wird häufig in Verbindung mit `-D` oder `-r` verwendet, um sicherzustellen, dass der `Checkout` in jedem Fall erfolgreich ist.

* `-j REV[:DATUM] oder -j REV1[:DATUM] -j REV2[:DATUM]`
    * Führt zwei Entwicklungslinien zu einer zusammen. Diese Option entspricht der `-j`-Option zu `update`, die wesentlich häufiger Verwendung findet (siehe `update` für weitere Details).

* `-k METHODE`
    * Setzt die Methode zur Schlüsselwortersetzung für RCS auf METHODE (so kann die standardmäßig eingestellte Methode für die behandelten Dateien verändert werden). Siehe auch den Abschnitt zur Schlüsselwortersetzung weiter hinten in diesem Kapitel. Die gewählte Methode wird permanent an die betroffenen Dateien gebunden - spätere `Updates` der Arbeitskopie werden sie beibehalten.

* `-l`
    * Lokal; führt lediglich einen `Checkout` auf die Hauptentwicklungslinie des Projektes aus. Unterverzeichnisse werden nicht behandelt.

* `-N`
    * Unterdrückt das Weglassen leerer Unterverzeichnisse mit der `-d`-Option.

* `-n`
    * Führt keines der `Checkout`-Programme aus, die in `CVSROOT/modules` mit der `-o`-Option angegeben sind. (Siehe auch den Abschnitt **Archivverwaltungsdateien** weiter hinten in diesem Kapitel für weitere Informationen.)

* `-P`
    * Löscht leere Verzeichnisse in der Arbeitskopie (entsprechend der `-P`-Option zu `update`).

* `-p`
    * Führt einen `Checkout` auf die Standardausgabe durch, nicht in Dateien (entsprechend der `-p`-Option zu `update`).

* `-R`
    * Behandelt auch rekursiv Unterverzeichnisse (dies ist der Normalfall). Siehe auch die `-f`-Option.

* `-r MARKE`
    * Führt einen `Checkout` der mit MARKE bezeichneten Revision durch. (Es würde kaum Sinn haben, hier eine numerische Revision (REV) zu verwenden, möglich ist es dennoch!) Diese Option bindet die mit MARKE bezeichnete Revision und impliziert `-P`.

* `-s`
    * Wie `-c`, zeigt aber den Status jedes Moduls und sortiert nach Status. (Siehe auch `CVSROOT/modules` im Abschnitt **Archivverwaltungsdateien**.)

5.5 commit [OPTIONEN] [DATEIEN]

Alternativen: ci, comm
Erfordert: Arbeitskopie, Archiv
Ändert: Archiv (und Verwaltungsdateien der Arbeitskopie)

Überträgt Änderungen einer Arbeitskopie in das Archiv.

Optionen:

* `-F MITTEILUNGSDATEI`
  * Benutzt den Inhalt der MITTEILUNGSDATEI für den Log-Eintrag, anstatt einen Editor zu aktivieren. Diese Option kann nicht mit `-m` kombiniert werden.

* `-f`
  * Erzwingt die Aktualisierung des Archivs mit einer neuen Revision, auch wenn keinerlei Änderungen in der Arbeitskopie gemacht wurden. `Commit` arbeitet mit dieser Option nicht rekursiv (d.h. es impliziert `-l`). Rekursion kann mit der Option `-R` erzwungen werden.

Bemerkung:

> Diese Bedeutung von `-f` stimmt mit der sonst üblichen Verwendung (**erzwinge die Verwendung der ersten Revision**) in den CVS-Kommandos nicht überein!

* `-l`
  * Lokal; überträgt nur Änderungen aus dem aktuellen Verzeichnis. Unterverzeichnisse werden nicht berücksichtigt.

* `-m MITTEILUNG`
  * Verwendet MITTEILUNG als Log-Dateieintrag, anstatt einen Editor zu aktivieren. Diese Option kann nicht mit `-F` kombiniert werden.

* `-n`
  * Startet kein Modulprogramm. (Siehe auch den Abschnitt **Archivverwaltungsdateien** in diesem Kapitel für weitere Informationen.)

* `-R`
  * Überträgt Änderungen auch aus Unterverzeichnissen. Dies ist das normale Verhalten, und die Option findet nur zur Aufhebung von `-l` in .cvsrc-Dateien Verwendung.

* `-r REV`
  * Überträgt Änderungen der Arbeitskopie in die Revision REV, die entweder ein Entwicklungszweig sein muss oder eine Revision, die höher als alle bisherigen Revisionen ist. `Commits` auf einer abgezweigten Version werden immer an das Ende angehängt - Sie können eine früher übertragene Version nicht **aktualisieren**. Die Nutzung dieser Option bindet die Revision fest an die der übertragenen Dateien. Dies kann mit `update -A` gelöscht werden. Die `-r`-Option impliziert `-f`. Eine neue Revision wird auch dann übertragen, wenn keine Dateien in der Arbeitskopie verändert wurden.

5.6 diff [OPTIONEN] [DATEIEN]

Alternativen: di, dif
Erfordert: Arbeitskopie, Archiv
Ändert: Nichts

Zeigt die Unterschiede zwischen zwei Revisionen (im Unix `diff`-Format) an. Wenn `diff` ohne weitere Optionen aufgerufen wird, dann zeigt CVS die (möglicherweise noch nicht durch einen `Commit` abgeglichenen) Unterschiede zwischen den Basisrevisionen im Archiv und der Arbeitskopie an. Die **Basis**-Revisionen bezeichnen dabei die letzten Revisionen, die aus dieser Arbeitkopie heraus vom Archiv bezogen wurden. Beachten Sie, dass es durchaus neuere Revisionen im Archiv geben könnte, nämlich dann, wenn jemand anderes zwischenzeitlich einen `Commit` irgendwelcher Änderungen vorgenommen hat, die noch nicht mit einem `Update` in diese Arbeitskopie übernommen wurden.

Optionen:

* `-D DATUM`
  * Erzeugt ein `Diff` gegen die neueste Revision, die nicht älter als DATUM ist. Diese Option verhält sich wie `-r` REV, außer dass statt der Revision ein Datum Verwendung findet.

* `-k METHODE`
  * Expandiert RCS-Schlüsselwörter in den `Diffs` unter Verwendung der angegebenen METHODE. (Siehe auch den Abschnitt **Schlüsselwortersetzung** weiter hinten in diesem Kapitel.)

* `-l`
  * Lokal; vergleicht nur Dateien aus dem aktuellen Verzeichnis. Unterverzeichnisse werden nicht berücksichtigt.

* `-r REV oder -r REV1 -r REV2`
  * Vergleicht entweder die aktuelle mit der angegebenen oder die beiden angegebenen Revisionen miteinander. Mit einer `-r`-Option wird die mit REV bezeichnete Revision aus dem Archiv mit der aktuellen Arbeitskopie verglichen. Bei Angabe zweier `-r`-Argumente wird der Unterschied zwischen den jeweils angegebenen Revisionen aus dem Archiv ermittelt. Im letzteren Fall spielt die Arbeitskopie keine Rolle, und die Revisionen können in beliebiger Reihenfolge angegeben werden - die Ausgabe spiegelt dann die **Richtung** des Vergleichsvorganges wider. Wird `-r` ganz weggelassen, so wird zwischen der Arbeitskopie und der Revision, worauf letztere basiert, verglichen.

5.7 Diff-Kompatibilitätsoptionen

Zusätzlich zu den angegebenen Optionen verwendet `cvs diff` noch einige Optionen, die mit der GNU-Version des normalen Kommandozeilenprogrammes `diff` übereinstimmen. Es folgt eine komplette Liste dieser Optionen, zusammen mit einer Erklärung einiger der üblicherweise am häufigsten verwendeten. (Für weitere Informationen siehe auch die GNU `diff`-Dokumentation.)

```
--0 -1 -2 -3 -4 -5 -6 -7 -8 -9
--binary
--brief
--changed-group-format=ARG
-c
-C NLINES
--context[=LINES]
-e --ed
```

```
-t --expand-tabs
-f --forward-ed
--horizon-lines=ARG
--ifdef=ARG
-w --ignore-all-space
-B --ignore-blank-lines
-i --ignore-case
-I REGEXP
--ignore-matching-lines=REGEXP
-h
-b --ignore-space-change
-T --initial-tab
-L LABEL
--label=LABEL
--left-column
-d --minimal
-N --new-file
--new-line-format=ARG
--old-line-format=ARG
--paginate
-n --rcs
-s --report-identical-files
-p
--show-c-function
-y --side-by-side
-F REGEXP
--show-function-line=REGEXP
-H --speed-large-files
--suppress-common-lines
-a --text
--unchanged-group-format=ARG
-u
-U NLINES
--unified[=LINES]
-V ARG
-W COLUMNS
--width=COLUMNS
```

Folgende GNU `diff`-Optionen werden am häufigsten mit cvs `diff` benutzt:

* `-B`
  * Ignoriert Unterschiede, die auf das Einfügen oder Löschen leerer Zeilen hinauslaufen.

* `-b`
  * Ignoriert Unterschiede in der Anzahl von Leerzeichen. Diese Option behandelt alle Folgen von Leerzeichen gleichwertig und ignoriert Leerzeichen am Zeilenende. Technischer ausgedrückt komprimiert diese Option alle Folgen von Leerzeichen auf je ein einziges und löscht Leerzeichen vom Zeilenende, bevor ein Vergleich stattfindet (siehe auch `-w`).

* `-c`
  * Erzeugt Ausgaben im Kontext, d.h. üblicherweise werden um jeden aufgelisteten Unterschied drei Zeilen des umgebenden Quelltextes (eben der Kontext) angezeigt. (Dies wird gemacht, damit das patch-Programm

verwendet werden kann, das mindestens zwei Zeilen Kontext benötigt.)

* `-C ANZ oder --context ANZ`
  * Wie `-c`, aber mit ANZ Zeilen an Kontext.

* `-i`
  * Der Vergleich nimmt mit `-i` keine Rücksicht auf Groß- und Kleinschreibung.

* `-u`
  * Zeigt die Ausgabe im so genannten unified `diff`-Format an.

* `-w`
  * Ignoriert alle Unterschiede bei Leerzeichen, sogar dann, wenn in der einen Datei an einer Stelle Leerzeichen sind, wo in der anderen überhaupt keine sind. Im Grunde eine verschärfte Version von `-b`.

5.7.1 edit [OPTIONEN] [DATEIEN]

Alternativen: Keine
Erfordert: Arbeitskopie, Archiv
Ändert: Zugriffsrechte in der Arbeitskopie, Watchliste im Archiv

Signalisiert, dass Sie dabei sind, beobachtete Datei(en) zu bearbeiten. Sie werden außerdem als temporärer Beobachter der Beobachterliste der Datei hinzugefügt. (Wenn Sie cvs unedit ausführen, werden Sie von der Liste entfernt; siehe auch `watch`, `watchers`, **unedit** und **editors**.)

Optionen:

* `-a AKTIONEN`
  * Legt fest, für welche Aktionen Sie temporärer Beobachter sein möchten. AKTIONEN sollten entweder edit, unedit, `commit`, all oder none sein. (Bei Nichtbenutzung von `-a` wird automatisch all angenommen.)

* `-l`
  * Lokal; signalisiert das Bearbeiten nur für Dateien im aktuellen Verzeichnis.

* `-R`
  * Rekursiv (dies ist die Standardeinstellung). Gegenteil von `-l`. `-R` nutzt nur, um ein `-l` in der `.cvsrc`-Datei rückgängig zu machen.

5.7.2 editors [OPTIONEN] [DATEIEN]

Alternativen: Keine
Erfordert: Arbeitskopie, Archiv
Ändert: nichts

Zeigt an, wer im Moment eine beobachtete Datei editiert (siehe auch `watch`, `watchers`, `unedit` und `edit`).

Optionen:

* `-l`
  * Lokal; Anzeige nur für Dateien im aktuellen Verzeichnis.

* `-R`
  * Rekursiv (dies ist die Standardeinstellung). Sie könnten `-R` benötigen, um ein `-l` in einer `.cvsrc`-Datei rückgängig zu machen.

5.7.3 export [OPTIONEN] PROJEKT(E)

Alternativen: exp, ex
Erfordert: Archiv
Ändert: Aktuelles Verzeichnis

Exportiert Dateien aus dem Archiv, um einen Projektbaum zu erzeugen, der keine Arbeitskopie darstellt (d.h. alle `CVS/`-Verwaltungsdateien fehlen). Hauptsächlich nützlich, um ein Distributionsarchiv zusammenzustellen.

Optionen:

* `-D DATUM`
  * Exportiert die neueste Revision, die nicht älter ist als DATUM.

* `-d VERZ`
  * Exportiert in das Verzeichnis VERZ (andernfalls wird der Verzeichnisname aus dem Modulnamen des Projekts genommen).

* `-f`
  * Erzwingt die Verwendung der ersten Revision, falls eine gegebene Bezeichnung oder Revision nicht gefunden werden kann (zur Nutzung mit `-D` oder `-r`).

* `-k METHODE`
  * Expandiert RCS-Schlüsselwörter nach der angegebenen METHODE. (Siehe den Abschnitt über Schlüsselwortersetzung weiter hinten in diesem Kapitel.)

* `-l`
  * Lokal; exportiert nur das Hauptverzeichnis, keine Unterverzeichnisse.

* `-N`
  * Verhindert das Weglassen leerer Unterverzeichnisse. Entspricht der Option `-N` beim `checkout`.

* `-n`
  * Startet keine modulspezifischen Programme, wie sie in CVSROOT/modules angegeben werden können. (Siehe auch den Abschnitt **Archivverwaltungsdateien** in diesem Kapitel für weitere Informationen.)

* `-P`
  * Leere Verzeichnisse werden weggelassen (wie die `-P`-Option bei `checkout` oder `update`).

* `-R`
  * Rekursiv (dies ist die Standardeinstellung). Alle Unterverzeichnisse des aktuellen Verzeichnisses werden exportiert. Sie könnten `-R` benötigen, um ein `-l` in einer `.cvsrc`-Datei rückgängig zu machen.

* `-r REV`
  * Exportiert die Revision REV. REV ist fast immer ein Name, keine numerische Revision.

5.7.4 gserver

Dies ist der GSSAPI-(Generic Security Services API-)Server. Dieses Kommando wird normalerweise nicht direkt von Benutzern ausgeführt. Es wird hingegen serverseitig gestartet, wenn ein Benutzer von einem Gastrechner mit der :gserver:-Zugriffsmethode eine Verbindung aufbaut:

```
user@linux ~/ # cvs -d :gserver:jrandom@floss.red-bean.com:/usr/local/repos
checkout projekt
```

Bemerkung:

> GSSAPI verwendet, unter anderem, die Kerberos-Version 5; für die Kerberos-Version 4 verwenden Sie bitte die :kserver:-Methode.
>
> Die Konfiguration und Benutzung einer GSSAPI-Bibliothek auf Ihren Maschinen ist jenseits des Umfanges dieses Buches. (Für weitere Hinweise siehe auch den Abschnitt **GSSAPI Authenticated** im Cederqvist.)

Optionen: Keine

5.7.5 history [OPTIONEN] [DATEINAMEN_BESTANDTEIL(E)]

Alternativen: hi, his
Erfordert: Archiv, CVSROOT/history
Ändert: Nichts

Zeigt eine zeitlich geordnete Auflistung der Aktivitäten im Archiv. Genauer gesagt zeigt diese Option eine Auflistung der Verwendung aller folgenden Kommandos: `checkout`, `commit`, `rtag`, `update` und `release`. Standardeinstellung ist das Auflisten von `Checkouts` (siehe aber die Option `-x`). Dieses Kommando funktioniert nicht, wenn die Datei `CVSROOT/history` im Archiv nicht existiert.

Das `history`-Kommando unterscheidet sich von anderen CVS-Kommandos auf mehrere Arten. Erstens benötigt es üblicherweise unbedingt Optionen, um etwas Sinnvolles zu tun (und einige dieser Optionen bedeuten hier etwas völlig anderes als bei den anderen CVS-Kommandos üblich). Zweitens nimmt es anstatt ganzer Dateinamen auch einen oder mehrere Wortbestandteile und listet dann alle Dateien auf, in deren Namen mindestens einer dieser Bestandteile vorkommt. Drittens sieht die Ausgabe von `history` ziemlich nach zufälligem Buchstabenmüll aus, solange man nicht gelernt hat, sie richtig zu interpretieren - daher wird das Ausgabeformat in einem eigenen Abschnitt im Anschluss an die Optionen erklärt (siehe auch **log**).

Optionen:

* `-a`
  * Zeigt eine Liste für alle Benutzer (normalerweise werden nur die eigenen Aktivitäten gelistet).

* `-b ZEICHENKETTE`

    * Listet alles bis zum ersten Eintrag, der ZEICHENKETTE im Modulnamen, Dateinamen oder Archivpfad enthält.

* `-c`
    * Commits werden mit aufgelistet.

* `-D DATUM`
    * Zeigt alle Daten seit DATUM (die üblichen CVS-Datumsformate sind anwendbar).

* `-e`
    * Alle Arten von Einträgen werden gelistet.

* `-f DATEI`
    * Zeigt die letzte aufgezeichnete Aktion an, die auf der Datei DATEI ausgeführt wurde. Diese Option kann mehrfach verwendet werden. Beachten Sie, dass hier `-f` eine andere Wirkung hat als bei den übrigen CVS-Kommandos: **Erzwinge neuste Revision, wenn alles andere fehlschlägt.**

* `-l`
    * Letztes; listet den jeweils letzten (neuesten) Eintrag jedes Projektes. Beachten Sie, dass hier `-l` eine andere Wirkung hat als bei den übrigen CVS-Kommandos: **Lokal ausführen, keine Unterverzeichnisse.**

* `-m MODUL`
    * Dies erzeugt einen kompletten Report über MODUL (ein Projektname). Diese Option kann mehrfach verwendet werden.

* `-n MODUL`
    * Listet das letzte (aktuellste) Ereignis, das für ein MODUL aufgezeichnet wurde. So ist, z.B. der `Checkout` eines Moduls ein Modulereignis, aber das Ändern oder Aktualisieren einer Datei innerhalb des Moduls ist ein Dateiereignis. Diese Option kann mehrfach verwendet werden. Beachten Sie, dass hier `-n` eine andere Wirkung hat als bei den übrigen CVS-Kommandos: **Keine `CVSROOT/modules`-Modulprogramme starten.**

* `-o`
    * Listet `Checkout`-Einträge (dies ist die Standardeinstellung).

* `-p ARCHIV`
    * Zeigt die Daten für ein bestimmtes Verzeichnis im Archiv. Diese Option kann mehrfach verwendet werden. Beachten Sie, dass hier `-p` eine andere Wirkung hat als bei den übrigen CVS-Kommandos: **Daten auf die Standardausgabe anstatt in eine Datei schreiben.**

Bemerkung:

Diese Option scheint zumindest teilweise fehlerhaft zu sein (Stand Sommer 1999).

* `-r REV`
    * Listet Einträge, die sich auf Revisionen nach der ersten Revision beziehen, ab der die Revision oder Marke mit dem Namen REV in einzelnen RCS-Dateien zu finden ist. Jede RCS-Datei wird dazu nach REV durchsucht.

* `-T`
  * Listet alle Markierungsvorgänge.

* `-t MARKE`
  * Zeigt alle Einträge, seit MARKE das letzte Mal zur`history`-Datei hinzugefügt wurde. Im Unterschied zur `-r`-Option wird hier nur die Datei`CVSROOT/history` durchsucht, und die Option ist daher wesentlich schneller.

* `-u BENUTZER`
  * Zeigt alle Vorfälle, die mit einem BENUTZER in Zusammenhang stehen. Diese Option kann mehrfach verwendet werden.

* `-w`
  * Zeigt alle Einträge, die sich auf das Verzeichnis beziehen, aus dem Sie das `history`-Kommando aufrufen.

* `-X HISTORY-DATEI`
  * Benutzt HISTORY-DATEI anstatt `CVSROOT/history`. Diese Option dient hauptsächlich zur Fehlersuche und wird nicht offiziell unterstützt; Sie könnten sie trotzdem nützlich finden (z.B. um Reports aus alten `history`-Dateien zu erzeugen, die Sie aufgehoben haben).

* `-x ARTEN`
  * Listet Vorfälle der angegebenen ARTEN auf. Jede Art von Eintrag wird durch einen Buchstaben aus der Liste **TOEFWUCGMAR** repräsentiert; beliebige Kombinationen aus diesen Buchstaben sind erlaubt. Ihre Bedeutung ist:
  * **T** MARKE
  * **O** Checkout
  * **E** Export
  * **F** Release
  * **W** Aktualisierung (bedeutungslos gewordene Datei aus Arbeitskopie gelöscht)
  * **U** Aktualisierung (Datei wurde durch Checkout überschrieben)
  * **C** Aktualisierung (Dateien wurden kombiniert, Konflikte traten auf)
  * **G** Aktualisierung (Dateien wurden kombiniert, keine Konflikte)
  * **M** Commit (Datei wurde verändert)
  * **A** Commit (Datei wurde hinzugefügt)
  * **R** Commit (Datei wurde entfernt)
  * Die Standardeinstellung, falls `-x` nicht angegeben wird, ist `Checkouts` anzuzeigen (wie `-x` O).

* `-z ZONE`
  * Listet Zeiten in Übereinstimmung mit der Zeitzone ZONE auf. ZONE ist eine Abkürzung für die gewünschte Zeitzone wie zum Beispiel UTC, GMT, BST, CET.1 Eine vollständige Liste aller Zeitzonen ist in der Datei `lib/getdate.c` in der CVS Quelltext-Distribution enthalten.

5.8 History-Ausgabe

Die Ausgabe des `history`-Befehls ist eine Folge von Zeilen; jede Zeile stellt ein Ereignis aus der `History`-Datei dar und beginnt mit einem einzelnen Symbolbuchstaben, der die Art des Ereignisses signalisiert. Zum Beispiel:

```
user@linux ~/ # cvs history -D yesterday -x TMO
```

```
M 08/21 20:19 +0000 jrandom 2.2 baar myproj == <remote>
M 08/22 04:18 +0000 jrandom 1.2 README myproj == <remote>
O 08/22 05:15 +0000 jrandom myproj =myproj= ~/src/*
M 08/22 05:33 +0000 jrandom 2.18 README.txt myproj == ~/src/myproj
O 08/22 14:25 CDT jrandom myproj =myproj= ~/src/*
O 08/22 14:26 CDT jrandom [99.08.23.19.26.03] myproj =myproj= ~/src/*
O 08/22 14:28 CDT jrandom [Exotic_Greetings-branch] myproj =myproj= ~/src/*
```

Die Symbolbuchstaben sind dieselben wie bei der obigen `-x`-Option. Auf den Buchstaben folgt das Datum des Ereignisses (in der Zeitzone UTC/GMT, falls nicht `-z` verwendet wurde), gefolgt von dem Namen des Benutzers, der für das Ereignis verantwortlich ist.

Hinter dem Benutzer kann eine Revisionsnummer, eine Marke oder ein Datum stehen, aber nur falls es dem entsprechenden Ereignis angemessen ist. (Datum oder Marke stehen in eckigen Klammern und sind wie oben gezeigt formatiert.) Beim `Commit` einer Datei steht dort die neue Revisionsnummer; wenn Sie mit `-D` oder `-r` einen `Checkout` machen, dann wird das gebundene Datum oder die angegebene Marke in eckigen Klammern angezeigt. Bei einem einfachen `Checkout` wird gar nichts angezeigt.

Als Nächstes kommt der Name der Datei, auf die sich der Eintrag bezieht bzw. der Name des Moduls, falls sich das Ereignis auf ein Modul bezieht. Im ersteren Fall folgen als Nächstes der Modul-/Projektname und der Name und das Verzeichnis der Arbeitskopie im Home-Verzeichnis des Benutzers. Im letzteren Fall sind die nächsten zwei Dinge, die auf das Modul folgen, der Name der ausgecheckten Arbeitskopie des Moduls (zwischen zwei Gleichheitszeichen), gefolgt von deren Standort im Home-Verzeichnis des Benutzers. (Der Name des Moduls in der Arbeitskopie kann von dem im Archiv abweichen, falls die Option `-d` beim `Checkout` verwendet wurde.)

5.8.1 import [OPTIONEN] [ARCHIV] [EXTERNE MARKE] [RELEASE_MARKE]

Alternativen: im, imp
Erfordert: Archiv, aktuelles Verzeichnis (das Quellverzeichnis)
Ändert: Archiv

Fügt neuen Quelltext in das Archiv ein, wobei entweder ein neues Projekt oder ein neuer Entwicklungszweig eines existierenden Projekts erzeugt wird. (Für eine grundlegende Erklärung von Entwicklungszweigen in `import` siehe auch Kapitel 6, das hilft Ihnen das Folgende zu verstehen.)

Es ist normal, `import` dazu zu verwenden, viele Dateien oder Verzeichnisse auf einen Schlag einem Projekt hinzuzufügen oder um ein neues Projekt zu kreieren. Um einzelne Dateien hinzuzufügen, sollten Sie `add` verwenden.

Optionen:

* `-b ZWEIG`
  * Importiert in den Entwicklungszweig ZWEIG. (ZWEIG ist eine echte Verzweigungsnummer, keine Marke.) Diese Option wird selten genutzt, kann aber nützlich sein, wenn man Quelltexte für dasselbe Projekt von unterschiedlichen Lieferanten erhält. Ein normales import-Kommando nimmt an, dass die Quelltexte in den Standardentwicklungszweig für externe Quelltexte importiert werden sollen, der **1.1.1** ist. Weil das üblich ist, macht sich meist keiner die Mühe, `-b` zu verwenden.

```
user@linux ~/ # cvs import -m "import von Lieferant 1" derenprojekt MARKE1
MARKE1-0
```

* * Um einen anderen Entwicklungszweig als den normalen zu importieren, müssen Sie die Nummer des Zweiges

explizit angeben:

```
user@linux ~/ # cvs import -b 1.1.3 -m "von Lieferant 2" derenprojekt MARKE2
MARKE2-0
```

* * Der 1.1.3-Zweig kann, wie jeder andere auch, weitere zukünftige Imports aufnehmen und ebenfalls wie jeder andere in die Hauptentwicklungslinie integriert werden. Dabei müssen Sie jedoch darauf achten, dass alle zukünftigen Imports, die `-b` 1.1.3 verwenden, auch dieselbe Marke (MARKE2) benutzen. CVS prüft nicht, ob die Nummer und die Marke einer abgezweigten Version übereinstimmen. Sollte dies jedoch nicht der Fall sein, werden unschöne und nicht vorhersagbare Dinge passieren.

Bemerkung:

Entwicklungszweige von externen Lieferanten [**Vendor Branches**, Anm.d.Übers.] werden normalerweise ungerade nummeriert, interne hingegen gerade.

* `-d`
  * Vermerkt den Zeitpunkt der letzten Änderung der Quelldateien anstatt der aktuellen Zeit als Zeitpunkt des Imports. Dies funktioniert nicht mit Client/Server-CVS.

* `-I NAME`
  * Dateien mit dem Namen NAME werden beim Import ignoriert. Diese Option kann mehrfach verwendet werden. Wildcards werden unterstützt: Bei Angabe von `*.foo` werden alle Dateien ignoriert, die auf `.foo` enden. (Für weitere Details zu Wildcards siehe auch `CVSROOT/cvsignore` im Abschnitt **Archivverwaltungsdateien**.) Die folgenden Datei- und Verzeichnisnamen werden standardmäßig ignoriert:

```
.
..
.#*
#*
,*
_$*
*~
*$
*.a
*.bak
*.BAK
*.elc
*.exe
*.ln
*.o
*.obj
*.olb
*.old
*.orig
*.rej
*.so
*.Z
.del-*
.make.state
.nse_depinfo
```

core
CVS
CVS.adm
cvslog.*
RCS
RCSLOG
SCCS
tags
TAGS

* * Sie können das Ignorieren dieser sowie eventuell weiterer, in `.cvsignore`, `CVSROOT/cvsignore` oder der Umgebungsvariablen CVSIGNORE aufgelisteter Dateinamen unterdrücken, indem Sie die Option `-I` ! verwenden. Das heißt,

```
user@linux ~/ # cvs import -I ! -m "importiere das Universum" proj MARKE
MARKE_0
```

* * importiert alle Dateien im aktuellen Verzeichnisbaum, auch solche, die normalerweise ignoriert werden würden.
  * Die Benutzung von `-I` ! löscht jede eventuell vorher erstellte Liste von zu ignorierenden Dateien, auch vorher angegebene `-I`-Optionen. Alle darauf folgenden `-I`-Optionen werden jedoch berücksichtigt. Daher ist

```
user@linux ~/ # cvs import -I ! -I README.txt -m "Mitteilung" projekt BLA
BLA_0
```

* * nicht dasselbe wie

```
user@linux ~/ # cvs import -I README.txt -I ! -m "Mitteilung" projekt BLA
BLA_0
```

* * Im ersten Fall wird `README.txt` nicht importiert, im letzteren Fall schon.

* `-k METHODE`
  * Legt die Methode zur RCS-Schlüsselwortersetzung für die importierten Dateien fest. (Für eine Liste gültiger Methoden siehe auch **Schlüsselwortersetzung** später in diesem Kapitel.)

* `-m MITTEILUNG`
  * Zeichnet die MITTEILUNG als die Import-Log-Mitteilung auf.

* `-W FILTER`
  * Spezifiziert dateinamenbasierte Filter, die für import gültig sein sollen. Diese Option kann mehrfach verwendet werden. (Zu Details über diese Filter siehe auch **CVSROOT/cvswrappers** im Abschnitt **Archivverwaltungsdateien**.)

5.8.2 init NEUES_ARCHIV

Alternativen: Keine

Erfordert: Einen Platz für das neue Archiv
Ändert: Archiv

Erzeugt ein neues Archiv (also ein Hauptarchiv, in dem dann viele verschiedene Projekte abgelegt werden können). Fast immer werden Sie hier die globale Option `-d` verwenden wollen, so wie in

```
user@linux ~/ # cvs -d /usr/local/neues_archiv init
```

denn selbst wenn Sie eine CVSROOT-Umgebungsvariable gesetzt haben, zeigt diese wahrscheinlich auf ein bereit existentes Archiv, was im Zusammenhang mit diesem Befehl nutzlos oder gar gefährlich sein könnte. (Zu weiteren Schritten, die nach der Initialisierung eines Archivs unternommen werden sollten, siehe Kapitel 4.)

Optionen: Keine

5.8.3 kserver

Dies ist der Kerberos-Server. (Falls Sie Kerberos-Bibliotheken Version 4 oder kleiner haben - Version 5 verwendet nur GSSAPI, siehe **gserver**.) Dieses Kommando wird normalerweise nicht direkt von Benutzern gestartet. Es wird hingegen serverseitig gestartet, wenn ein Benutzer von einem Gastrechner mit der :kserver: Zugriffsmethode eine Verbindung aufbaut:

```
user@linux ~/ # cvs -d :kserver:jrandom@floss.red-bean.com:/usr/local/repos
checkout projekt
```

Die Konfiguration und Benutzung von Kerberos auf Ihrer Maschine an dieser Stelle zu erläutern, würde den Umfanges dieses Buches überschreiten. (Für einige nützliche Hinweise siehe auch den Abschnitt **Kerberos Authenticated** im Cederqvist.)

Optionen: Keine

5.8.4 log [OPTIONEN] [DATEIEN]

Alternativen: lo, rlog
Erfordert: Arbeitskopie, Archiv
Ändert: Nichts

Zeigt Log-Einträge für ein Projekt oder Dateien innerhalb eines Projektes an. Die Ausgabe von log ist nicht gerade im selben Stil wie die Ausgaben anderer CVS-Kommandos, da log auf einem älteren RCS-Programm basiert (rlog). Sein Ausgabeformat enthält einen Kopfteil, in dem verschiedene, nicht revisionsspezifische Details über die Datei enthalten sind, gefolgt von den eigentlichen Log-Mitteilungen (geordnet nach Revision). Jede Revision enthält nicht nur die Revisionsnummer und eine Mitteilung, sondern auch den Autor und das Datum der Änderung sowie die Anzahl der hinzugefügten oder gelöschten Zeilen. Alle Zeitangaben werden in UTC (GMT) angegeben, nicht Lokalzeit.

Da log-Ausgaben dateiweise erfolgen, muss ein einzelner `Commit`, der mehrere Dateien umfaßt, nicht unbedingt auch in geschlossener Form erscheinen. Wenn Sie jedoch alle Log-Mitteilungen und Daten aufmerksam verfolgen, dann ist es möglich zu rekonstruieren, was vorging. (Für Informationen über ein Programm, das mehrere Log-Dateien zu einer wesentlich lesbareren Form zusammenfassen kann, siehe auch **cvs2cl.pl: die Erzeugung von Change-Logs im GNU-Stil aus CVS Logs** in Kapitel 10; siehe auch `history`.)

Optionen:

Wenn Sie die folgenden Filteroptionen betrachten, wird möglicherweise nicht auf Anhieb klar, wie diese sich verhalten, wenn man sie kombiniert. Eine exakte Beschreibung des Verhaltens von log ist die, dass es stets die Schnittmenge der mit `-d`, `-s` und `-w` selektierten Revisionen, geschnitten mit der Vereinigungsmenge der mit `-b` und `-r` gewählten,

anzeigt.

* `-b`
  * Gibt nur Log-Informationen über den Standardentwicklungszweig aus (üblicherweise die höchste Abzweigung in der Hauptentwicklungslinie). Diese Option wird üblicherweise benutzt, um die Ausgabe von Log-Mitteilungen aus Seitenzweigen der Entwicklung zu unterdrücken.

* `-dDATUM`
  * Selektiert Log-Informationen für die Revisionen, deren Datum zu dem Datum oder zu dem Datumsbereich passt, der durch DATUM gegeben ist, einer durch Semikola separierten Liste. Die Datumsangaben können in jedem der üblichen Datumsformate gehalten sein (siehe **Datumsformate** weiter vorne in diesem Kapitel) und können wie folgt in Bereiche kombiniert werden:

* `DATUM1<DATUM2`
  * Bezeichnet Revisionen, die zwischen DATUM1 und DATUM2 entstanden sind. Falls DATUM1 zeitlich auf DATUM2 folgt, sollte > anstatt von < verwendet werden; sonst bleibt die Anweisung wirkungslos.

* `<DATUM DATUM>`
  * Alle Revisionen vonDATUM oder früher.

* `<DATUM DATUM>`
  * Alle Revisionen vonDATUM oder später.

* `DATUM`
  * Bezeichnet die einzige aktuellste Version vom DATUM oder früher.

Sie können <= und >= anstatt von < und > verwenden, um einen inklusiven Bereich zu definieren (ansonsten sind die Bereiche exklusiv). Mehrere Bereiche können durch Semikola getrennt definiert werden, so selektiert zum Beispiel

```
user@linux ~/ # cvs log -d"199-06-01<1999-07-01;1999-08-01<1999-09-01"
```

alle Log-Mitteilungen für Revisionen, deren `Commit` im Juni oder August 1999 durchgeführt wurde (wobei Juli ausgelassen wird). Es dürfen keine Leerzeichen zwischen `-d` und seinen Argumenten stehen.

* `-h`
  * Gibt nur den Kopfteil der Information zu jeder Datei aus, der den Dateinamen, das Arbeitsverzeichnis, die aktuellste Revision, den Entwicklungszweig, die Zugriffsliste, Sperren (Locks), symbolische Namen (Marken) und die für diese Datei eingestellte Methode der Schlüsselwortersetzung enthält. Es werden keine Log-Mitteilungen ausgegeben.

* `-l`
  * Lokal; bezieht sich nur auf Dateien im aktuellen Verzeichnis.

* `-N`
  * Lässt die Liste der symbolischen Namen (Marken) aus dem Kopfteil weg. Dies kann nützlich sein, wenn Ihr Projekt viele Markierungen beinhaltet, Sie aber nur an den Log-Mitteilungen interessiert sind.

* `-R`
  * Gibt den Namen der RCS Datei in dem Archiv aus.

Bemerkung:

> Diese Bedeutung von `-R` unterscheidet sich von der sonst in CVS-Befehlen üblichen Funktion: **Rekursiv**. Daher gibt es keine Möglichkeit, ein in Ihrer `.cvsrc`-Datei für dieses Kommando vorhandenes `-l` aufzuheben. Schreiben Sie daher nicht log `-l` in Ihre `.cvsrc`-Datei.

* `-rREVS`
  * Zeigt Log-Informationen zu den in REVS spezifizierten Revisionen - eine durch Kommata separierte Liste. REVS kann sowohl Revisionsnummern als auch symbolische Namen enthalten. Bereiche können wie folgt definiert werden:
    * `REV1:REV2` Revisionen von REV1 bis REV2 (auf demselben Entwicklungszweig!)
    * `:REV` Revisionen vom Beginn der abgezweigten Version bis REV (inklusive)
    * `REV:` Revisionen von REV bis zum Ende der abgezweigten Version
    * `ZWEIG` Alle Revisionen einer abgezweigten Version von Anfang bis Ende
    * `ZWEIG1:ZWEIG2` Mehrere Entwicklungszweige - alle Revisionen auf allen Zweigen im angegebenen Bereich
    * `ZWEIG.` Die aktuellste Revision von ZWEIG

Zudem selektiert ein allein stehendes `-r` ohne Argument die neueste Revision des Standardzweiges (normalerweise der Hauptentwicklungslinie). Leerzeichen zwischen `-r` und seinem Argument sind nicht gestattet.

Bemerkung:

> Wenn das Argument zu `-r` eine Liste ist, muss diese durch Kommata getrennt sein, nicht durch Semikola wie in `-d`.

* `-sSTATUS`
  * Findet Revisionen, deren Statusattribut mit einem der in STATUS angegebenen Zustände übereinstimmt, eine durch Kommata separierte Liste. Zwischen `-s` und seinem Argument darf kein Leerzeichen sein.

Bemerkung:

> Wenn das Argument zu `-s` eine Liste ist, muss diese durch Kommata getrennt sein, nicht durch Semikola wie in `-d`.

* `-t`
  * Wie `-h`, beinhaltet aber zusätzlich die Dateibeschreibung (die beim ersten Erstellen der Datei entstand).

* `-wBENUTZER`
  * Findet Revisionen, deren `Commit` die Benutzer veranlasst haben, die in der durch Kommata separierten Liste BENUTZER aufgelistet sind. Ein `-w` ohne Argumente bewirkt eine Suche nach dem Benutzer, der cvs log gestartet hat. Bedenken Sie, dass, wenn Benutzer-Aliasing aktiviert ist, CVS den CVS-Benutzernamen anstatt des Systembenutzernamens beim `commit` aufzeichnet. Leerzeichen zwischen `-w` und seinem Argument sind nicht möglich.

Bemerkung:

> Wenn das Argument zu `-w` eine Liste ist, muss diese durch Kommata getrennt sein, nicht durch Semikola wie in

`-d`.

5.8.5 login

Alternativen: logon, lgn
Erfordert: Archiv
Ändert: ~/.cvspass-Datei

Kontaktiert einen CVS-Server und überprüft die Zugangsberechtigung für ein bestimmtes Archiv. Dieses Kommando beeinflusst weder das Archiv noch eine Arbeitskopie; es überprüft lediglich das Passwort (zur Nutzung mit der :pserver:-Zugriffsmethode) für ein Archiv und speichert das Passwort zur späteren Benutzung in der `.cvspass`-Datei in Ihrem Home-Verzeichnis. Zukünftige Kommandos, die auf dasselbe Archiv mit demselben Benutzernamen zugreifen, erfordern dann kein erneutes Starten von `login`, da Ihr CVS einfach die Datei`.cvspass` konsultiert, um das Passwort zu erfahren.

Wenn Sie dieses Kommando verwenden, dann sollten Sie ein Archiv unter Verwendung der pserver-Zugriffsmethode angeben, beispielsweise

```
user@linux ~/ # cvs -d :pserver:jrandom@floss.red-bean.com:/usr/local/archiv
```

oder durch Setzen der CVSROOT-Umgebungsvariablen.

Wenn sich das Passwort auf dem Server ändert, müssen Sie `login` erneut starten.

Optionen: Keine

5.8.6 logout

Alternativen: Keine
Erfordert: ~/.cvspass-Datei
Ändert: ~/.cvspass-Datei

Das Gegenteil von `login` - entfernt das Passwort für das angegebene Archiv aus der `.cvspass`-Datei.

Optionen: Keine

5.8.7 pserver

Alternativen: Keine
Erfordert: Archiv
Ändert: Nichts

Pserver ist der passwortauthentisierende Server. Dieser Befehl wird normalerweise nicht direkt von Benutzern ausgeführt. Er wird vielmehr durch /etc/inet.conf gestartet, wenn ein Benutzer von einem Gastrechner mit der :pserver:-Zugriffsmethode eine Verbindung aufbaut. (Siehe auch `login`, `logout` und `.cvspass` im Abschnitt **Laufzeit-Kontrolldateien** in diesem Kapitel. Für Details zur Konfiguration eines passwortauthentisierenden CVS-Servers siehe Kapitel 4.)

Optionen: keine

5.8.8 rdiff [OPTIONEN] PROJEKTE

Alternativen: patch, pa
Erfordert: Archiv

Ändert: Nichts

Wie das `diff`-Kommando, mit dem Unterschied, dass es direkt im Archiv arbeitet und daher keine Arbeitskopie erfordert. Dieses Kommando ist zur Ermittlung von Unterschieden zwischen verschiedenen Versionen Ihrer Projekte gedacht. Die Ausgabe erfolgt in einem Format, das als Eingabe für das patch-Programm verwendet werden kann (sodass Sie ggf. Patches für Benutzer verteilen können, die Ihre Programme aktualisieren möchten).

Die Benutzung des Programmes patch liegt außerhalb des Themenbereiches dieses Buches. Seien Sie trotzdem darauf hingewiesen, dass die Verwendung der Option `-p` für patch erforderlich sein kann, wenn die Patch-Datei Änderungen für Dateien in Unterverzeichnissen enthält, um patch dazu zu bringen, die Aktualisierung erfolgreich durchzuführen. (Siehe die Dokumentation zu patch für weitere Informationen; siehe auch `diff`.)

Optionen:

* `-c`
  * Erzeugt Ausgaben im context-`diff`-Format (Standardeinstellung).

* `-D DATUM oder -D DATUM1 -D DATUM2`
  * Mit einem Datum erzeugt diese Option die Unterschiede zwischen den Dateien zum Zeitpunkt DATUM und der aktuellsten Revision. Mit zwei Daten wird der Unterschied der Dateien zu den jeweiligen Daten festgestellt.

* `-f`
  * Erzwingt die Verwendung der ersten Revision, falls eine gegebene Revision mit `-D` oder `-r` nicht gefunden werden kann.

* `-l`
  * Lokal; vergleicht nur Dateien aus dem aktuellen Verzeichnis.

* `-R`
  * Rekursiv; Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

* `-r REV oder -r REV1 -r REV2`
  * Mit einer `-r`-Option wird die mit REV bezeichnete Revision aus dem Archiv mit der aktuellen Arbeitskopie verglichen. Bei Angabe zweier `-r`-Argumente wird der Unterschied zwischen den jeweilig angegebenen Revisionen aus dem Archiv ermittelt.

* `-s`
  * Zeigt eine Zusammenfassung der Unterschiede an. Es wird gezeigt, welche Dateien hinzugefügt, geändert oder entfernt wurden, ohne die konkreten Veränderungen am Inhalt aufzulisten. Die Ausgabe sieht so aus:

```
user@linux ~/ # cvs -Q rdiff -s -D 1999-08-20 myproj

File myproj/Random.txt is new; current revision 1.4
File myproj/README.txt changed from revision 2.1 to 2.20
File myproj/baar is new; current revision 2.3
```

* `-t`

    * Zeigt die Unterschiede zwischen den zwei aktuellsten Revisionen jeder Datei. Dies ist eine praktische Abkürzung zur Ermittlung der letzten Änderungen an einem Projekt. Diese Option kann nicht in Verbindung mit `-D` und `-r` verwendet werden.

* `-u`
    * Erzeugt Ausgabe im unified `diff`-Format. Ältere Versionen des Programmes patch können mit dem unified `diff`-Format nicht umgehen; darum verwenden Sie diese Option besser nicht, um einen distributionsfähigen Patch zu erzeugen; benutzen Sie hingegen `-c`.

* `-V (Überholt)`
    * CVS erzeugt eine Fehlermeldung, wenn Sie versuchen sollten, diese Option noch zu verwenden. Ich führe sie hier nur mit auf, falls Sie sie einmal in einem alten Skript finden sollten.

5.8.9 release [Optionen] VERZEICHNIS

Alternativen: re, rel
Erfordert: Arbeitskopie
Ändert: Arbeitskopie, CVSROOT/history

Meldet eine Arbeitskopie vom Archiv ab (markiert sie als nicht länger in Gebrauch). Im Gegensatz zu den meisten anderen CVS-Befehlen, die auf eine Arbeitskopie wirken, wird dieser Befehl nicht aus dem Verzeichnis der Arbeitskopie heraus aufgerufen, sondern aus dem direkt darüber liegenden (dem Mutterverzeichnis). Sie müssen entweder Ihre `CVSROOT`-Umgebungsvariable setzen oder die globale Option `-d` verwenden, da CVS sonst nicht in der Lage ist, das Archiv von der Arbeitskopie herauszufinden.

Die Benutzung von release ist nie erforderlich. Da CVS normalerweise keine Referenzen auf Arbeitskopien führt, können Sie Ihre Arbeitskopie auch einfach löschen.

Falls Sie jedoch Änderungen in Ihrer Arbeitskopie haben, von denen Sie noch keinen `Commit` durchgeführt haben, oder wenn Sie möchten, dass das Beenden Ihrer Arbeit als Ereignis in der `CVSROOT/history`-Datei vermerkt wird (siehe das `history`-Kommando), dann sollten Sie release verwenden.

Optionen:

* `-d`
    * Löscht die Arbeitskopie, wenn die Abmeldung erfolgreich war. Ohne `-d` bleibt die Arbeitskopie auf der Platte auch nach dem Release erhalten.

Bemerkung:

> Falls Sie irgendwelche neuen Verzeichnisse innerhalb Ihrer Arbeitskopie erzeugt haben, diese aber nicht dem Archiv hinzugefügt wurden, dann werden sie zusammen mit dem Rest der Arbeitskopie gelöscht, wenn Sie `-d` verwenden.

5.8.10 remove [OPTIONEN] [DATEIEN]

Alternativen: rm, delete
Erfordert: Arbeitskopie
Ändert: Arbeitskopie

Entfernt eine Datei aus einem Projekt. Normalerweise wird die Datei direkt von der Platte gelöscht, wenn Sie dieses

Kommando aufrufen (siehe jedoch `-f`). Obwohl dieser Befehl normalerweise rekursiv arbeitet, ist es üblich, die zu löschenden Dateien einzeln explizit zu benennen. Beachten Sie die Merkwürdigkeit im letzten Satz: Üblicherweise starten Sie cvs remove für Dateien, die in Ihrer Arbeitskopie gar nicht mehr existieren.

Obwohl das Archiv zur Bestätigung kontaktiert wird, werden die Dateien nicht wirklich gelöscht, bis später ein `commit` durchgeführt wird. Und selbst dann wird die RCS-Datei nicht wirklich aus dem Archiv gelöscht; wenn sie aus der Hauptentwicklungslinie stammt, wird sie lediglich in ein Attic/-Unterverzeichnis verschoben, wo sie noch immer für andere Entwicklungszweige zur Verfügung steht. Stammt sie aus einem Zweig, dann wird sie nicht bewegt, aber eine neue Revision mit dem Status **dead**2 auf der abgezweigten Version erzeugt (siehe auch add.)

Optionen:

* `-f`
  * Löscht die Datei vor dem Löschen im CVS zunächst von der Platte. Diese Bedeutung von `-f` unterscheidet sich von der sonst in CVS-Befehlen üblichen Funktion: **Erzwinge ggf. aktuellste Revision.**

* `-l`
  * Lokal; der Befehl bezieht sich nur auf das aktuelle Arbeitsverzeichnis.

* `-R`
  * Rekursiv; Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

5.8.11 rtag [OPTIONEN] MARKE PROJEKT(E)

Alternativen: rt, rfreeze
Erfordert: Archiv
Ändert: Archiv

Markiert ein Modul direkt im Archiv (erfordert keine Arbeitskopie). Sie sollten Ihre CVSROOT-Umgebungsvariable korrekt gesetzt haben oder die Option `-d` benutzen, damit dieses Kommando funktioniert (siehe auch tag).

Optionen:

* `-a`
  * Entfernt Markierungen von allen **gelöschten** Dateien, da diese lediglich aus historischen Gründen noch im Archiv vorhanden sind, nicht aber länger als Bestandteil des aktiven Projektes betrachtet werden. Obwohl es nicht erlaubt ist, Dateien mit einem symbolischen Namen zu markieren, der bereits an anderer Stelle verwendet wird, sollte es keine Probleme geben, falls diese Marke nur noch in **gelöschten** Dateien Verwendung findet (die aus der Sicht des aktuellen Projektes ja nicht mehr länger existent sind).

* `-b`
  * Erzeugt eine neue abgezweigte Version mit dem Namen MARKE.

* `-D DATUM`
  * Markiert die neuesten Revisionen, die nicht neuer als DATUM sind.

* `-d`
  * Löscht die Markierung mit dem symbolischen Namen MARKE. Diese Änderung wird nicht protokolliert - die Markierung verschwindet einfach. CVS führt keine Protokolle der Änderungen an Markierungen.

* `-F`
  * Erzwingt die Neuvergabe des symbolischen Namens, falls dieser bereits für irgendeine andere Revision in der Datei vergeben sein sollte.

* `-f`
  * Erzwingt die Verwendung der ersten Revision, falls eine gegebene Bezeichnung oder Revision nicht gefunden werden kann (siehe `-D` und `-r`).

* `-l`
  * Lokal; der Befehl bezieht sich nur auf das aktuelle Arbeitsverzeichnis.

* `-n`
  * Verhindert die Ausführung von markierungsabhängigen Programmen, die in CVSROOT/modules angegeben sein könnten. (Siehe auch den Abschnitt **Archivverwaltungsdateien** für weitere Details zu diesen Programmen.)

* `-R`
  * Rekursiv; Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

* `-r REV`
  * Markiert Revision REV. (Dabei kann REV wieder ein symbolischer Name sein.)

5.8.12 server

Startet einen CVS-Server. Dieser Befehl wird niemals von Benutzern ausgeführt (außer sie versuchen, Fehler im Client/Server-Protokoll zu finden), daher vergessen Sie einfach, dass ich ihn überhaupt erwähnt habe.

Optionen: Keine

5.8.13 status [OPTIONEN] [DATEIEN]

Alternativen: st, stat
Erfordert: Arbeitskopie
Ändert: Nichts

Zeigt den Status von Dateien in der Arbeitskopie an.

Optionen:

* `-l`
  * Lokal; der Befehl bezieht sich nur auf das aktuelle Arbeitsverzeichnis. Eventuelle Unterverzeichnisse werden nicht behandelt.

* `-R`
  * Rekursiv; Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

* `-v`
  * Zeigt symbolische Namen (Markierungen) für die Datei an.

5.8.14 tag [OPTIONEN] MARKIERUNG [DATEIEN]

Alternativen: ta, freeze
Erfordert: Arbeitskopie, Archiv
Ändert: Archiv

Gibt einer bestimmten Revision oder einer Reihe von Revisionen eines Projektes einen symbolischen Namen. Oft nennt man diesen Vorgang auch **eine Momentaufnahme des Projektes machen**. Dieser Befehl wird ebenfalls dazu verwendet, neue Entwicklungszweige in CVS zu erzeugen (siehe die `-b`-Option; siehe auch rtag).

Optionen:

* `-b`
  * Erzeugt eine neue abgezweigte Version mit dem Namen MARKE.

* `-c`
  * Stellt sicher, dass die Arbeitskopie keine Änderungen beinhaltet, die noch nicht mit einem `Commit` bestätigt wurden. Sollte dies der Fall sein, wird eine Warnung ausgegeben und keine Markierung erzeugt.

* `-D DATE`
  * Markiert die neuesten Revisionen, die nicht neuer als DATUM sind.

* `-d`
  * Löscht die Markierung mit dem symbolischen Namen MARKE. Diese Änderung wird nicht protokolliert - die Markierung verschwindet einfach. CVS führt keine Protokolle der Änderungen an Markierungen.

* `-F`
  * Erzwingt die Neuvergabe des symbolischen Namens, falls dieser bereits für irgendeine andere Revision in der Datei vergeben sein sollte.

* `-f`
  * Erzwingt die Verwendung der ersten Revision, falls eine gegebene Bezeichnung oder Revision nicht gefunden werden kann (siehe `-D` und `-r`).

* `-l`
  * Lokal; der Befehl bezieht sich nur auf das aktuelle Arbeitsverzeichnis.

* `-R`
  * Rekursiv; Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

* `-r REV`
  * Markiert Revision REV (welche selber ein symbolischer Name sein kann).

5.8.15 unedit [OPTIONEN] [DATEIEN]

Alternativen: Keine
Erfordert: Arbeitskopie, Archiv
Ändert: Arbeitskopie

Signalisiert Beobachtern, dass Sie mit dem Bearbeiten einer Datei fertig sind (siehe auch `watch`, `watchers`, `edit` und `editors`).

Optionen:

* `-l`
  * Lokal; signalisiert nur für Dateien im aktuellen Verzeichnis.

* `-R`
  * Rekursiv; Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

5.8.16 update [OPTIONEN] [DATEIEN]

Alternativen: up, upd
Erfordert: Arbeitskopie, Archiv
Ändert: Arbeitskopie

Überführt Änderungen aus dem Archiv in Ihre Arbeitskopie. Als Seiteneffekt wird angezeigt, welche Dateien in Ihrer Arbeitskopie verändert wurden. (Falls jedoch die globale Option `-Q` verwendet wird, unterbleibt diese Anzeige; siehe auch `checkout`.)

Optionen:

* `-A`
  * Löscht alle gebundenen Markierungen, Daten oder RCS-Schlüsselwortersetzungs-Methoden. Dies kann zu Veränderungen im Inhalt von Dateien führen, falls sich die aktuellsten Revisionen der Hauptentwicklungslinie von den bisher durch Bindungen festgelegten unterscheiden. (Sehen Sie `-A` einfach als einen neuen `Checkout` des Projekthauptzweiges an.)

* `-D DATUM`
  * Aktualisiert bis zur letzten Revision, die nicht älter als DATUM ist. Diese Option wirkt bindend und impliziert `-P`. Sollte das Datum der Arbeitskopie gebunden sein, dann sind keine `Commits` möglich.

* `-d`
  * Holt fehlende Verzeichnisse, also Verzeichnisse, die im Archiv existieren, aber in der Arbeitskopie noch nicht vorhanden sind. Solche Verzeichnisse könnten im Archiv angelegt worden sein, nachdem die Arbeitskopie zuletzt per `Checkout` übertragen wurde. Ohne diese Option arbeitet `update` nur auf Verzeichnissen, die bereits in der Arbeitskopie vorhanden sind; neue Dateien werden aus dem Archiv geholt, neue Verzeichnisse aber nicht (siehe auch `-P`).

* `-f`
  * Erzwingt die Verwendung der aktuellsten Revision, falls eine mittels `-D` oder `-r` gegebene Revision nicht

gefunden werden kann.

* `-I NAME`
  * Wie die `-I`-Option des Befehles import.

* `-j REV[:DATUM] oder -j REV1[:DATUM] -j REV2[:DATUM]`
  * Vereinigt zwei Entwicklungszweige. Wenn wir das optionale DATUM-Argument zunächst ignorieren (darauf wird später eingegangen), dann funktioniert `-j` so: Wenn nur ein `-j` gegeben ist, dann werden alle Veränderungen vom Zeitpunkt der Abspaltung des Entwicklungszweiges an bis zur Revision REV genommen und mit der Arbeitskopie vereinigt. Dabei entspricht der Zeitpunkt der Abspaltung der letzten Revision, die der Entwicklungszweig im Arbeitsverzeichnis und die abgezweigte Versionslinie von REV gemeinsam haben. Werden zwei `-j`-Optionen angegeben, werden die Veränderungen von REV1 bis REV2 in die Arbeitskopie integriert.
  * Die speziellen Bezeichner HEAD und BASE dürfen als Argumente für `-j` verwendet werden; sie bezeichnen die aktuellste Revision im Archiv respektive die Revision im Archiv, auf der die aktuelle Arbeitskopie basiert.
  * Was das optionale DATUM angeht: Falls sich REV auf eine abgezweigte Version bezieht, bezeichnet es normalerweise die neueste Revision auf diesem Zweig, Sie können jedoch diese auf die neueste Revision vor einem beliebigen DATUM beschränken. Das Datum sollte von der Revision durch einen Doppelpunkt ohne Leerzeichen abgetrennt sein wie in folgendem Beispiel:

```
user@linux ~/ # cvs update -j einZweig:1999-07-01 -j einZweig:1999-08-01
```

  * In diesem Beispiel werden unterschiedliche Datumsangaben auf demselben Entwicklungszweig verwendet, sodass hier im Endeffekt die Veränderungen auf dem Entwicklungszweig **einZweig** zwischen Juli und August genommen und in die Arbeitskopie integriert werden. Beachten Sie jedoch, dass der Entwicklungszweig nicht notwendigerweise in beiden `-j`-Optionen derselbe sein muss.

* `-k METHODE`
  * Verwendet RCS-Schlüsselwortersetzung gemäß der angegebenen METHODE. (Siehe auch den Abschnitt zur Schlüsselwortersetzung weiter hinten in diesem Kapitel.) Die gewählte Methode wirkt für die betroffenen Dateien bindend - spätere Aktualisierungen der Arbeitskopie werden davon betroffen (siehe aber `-A`).

* `-l`
  * Lokal: aktualisiert nur das aktuelle Arbeitsverzeichnis.

* `-P`
  * Leere Verzeichnisse werden gelöscht. Alle CVS-kontrollierten Verzeichnisse, die am Ende der Aktualisierung keine Dateien mehr enthalten, werden aus der Arbeitskopie entfernt (siehe auch `-d`).

* `-p`
  * Gibt die Dateiinhalte auf die Standardausgabe aus, anstatt in die Dateien zu schreiben. Wird hauptsächlich verwendet, um zu einer früheren Revision zurückzugehen, ohne Bindungen (Sticky Tags) in der Arbeitskopie zu produzieren. Zum Beispiel:

```
user@linux ~/ # cvs update -p -r 1.3 README.txt >README.txt
```

*

    * Nun hat README.txt wieder den Inhalt seiner früheren Revision 1.3, gerade so, als hätten Sie die Änderungen von Hand zurückgenommen.

* `-R`
    * Rekursiv; Unterverzeichnisse werden ebenfalls behandelt. Da dies das Standardverhalten ist, dient `-R` lediglich zur Änderung des Verhaltens der `-l`-Option in `.cvsrc`-Dateien.

* `-r REV`
    * Aktualisiert (bzw deaktualisiert oder verzweigt) auf Revision REV. Wenn eine komplette Arbeitskopie aktualisiert wird, ist REV entweder einfach eine Marke oder die Markierung eines Zweiges. Bei der Aktualisierung einer einzelnen Datei kann es sich jedoch genauso gut um eine Revisionsnummer handeln.
    * Diese Option wirkt bindend. Wenn die Dateien auf eine nicht zum aktuellen Entwicklungszweig gehörige Markierung oder Revision verändert werden, dann können Sie keinen `Commit` mehr durchführen, bis die Bindung aufgehoben wurde (siehe `-A`). Wenn REV der Name einer abgezweigten Version war, sind `Commits` jedoch möglich und neue Revisionen fließen in diesen Zweig ein.

* `-WFILTER`
    * Spezifiziert dateinamenbasierte Filter, die für das `Update` gültig sein sollen. Diese Option kann mehrfach verwendet werden. (Siehe auch `CVSROOT/cvswrappers` im Abschnitt **Archivverwaltungsdateien** zu Details über diese Filter.) Zwischen `-W` und seinem Argument ist kein Leerzeichen.

5.8.17 watch on|off|add|remove [OPTIONEN] [DATEIEN]

Alternativen: Keine
Erfordert: Arbeitskopie, Archiv
Ändert: Watchliste im Archiv

Schaltet die Beobachtung einer oder mehrerer Dateien ein. Anders als die meisten CVS-Befehle benötigt `watch` ein weiteres Subkommando, um etwas Sinnvolles zu bewirken. (Siehe auch `watchers`, `edit`, `editors`, `unedit` und `CVSROOT/users` im Abschnitt **Archivverwaltungsdateien** in diesem Kapitel.)

Subkommandos:

* `on`
    * Schaltet das Beobachten der Dateien ein. Das bedeutet, dass sie beim `Checkout` als nur lesbar markiert werden und Benutzer cvs edit benutzen sollten, um sie beschreibbar zu machen (wobei alle Beobachter darüber informiert werden, dass die Datei nun bearbeitet wird). Das Einschalten an sich bewirkt nicht, dass Sie zur Watchliste irgendeiner Datei hinzugefügt werden. (Siehe `watch add` und `watch remove` zu diesem Problem.)

* `off`
    * Das Gegenteil von `watch on`. Schaltet das Beobachten wieder ab.

* `add`
    * Fügt Sie der Liste der Beobachter dieser Datei (Watchliste) hinzu. Sie werden benachrichtigt, sobald jemand einen `Commit` dieser Datei vornimmt, `cvs edit` oder `cvs unedit` ausführt (siehe jedoch die `-a`-Option).

* `remove`

* Das Gegenteil von `watch add`. Entfernt Sie aus der Watchliste dieser Datei.
* Optionen (zur Benutzung mit jedem der `watch`-Subkommandos). Alle drei Optionen haben dieselbe Bedeutung wie für `edit`:

* `-a AKTIONEN`
* `-l`
* `-R`

5.8.18 watchers [OPTIONEN] [DATEIEN]

Alternativen: Keine
Erfordert: Arbeitskopie, Archiv
Ändert: Nichts

Zeigt an, wer Dateien beobachtet.

Optionen - Diese Optionen haben dieselbe Bedeutung wie für `edit`:

* `-l`
* `-R`

6 Schlüsselwortersetzung (RCS-Schlüsselwörter)

CVS kann bestimmte textuelle Ersetzungen in Dateien durchführen, die es Ihnen erlauben, einige Arten von Informationen automatisch innerhalb Ihrer Dateien auf dem aktuellsten Stand zu halten. Alle diese Ersetzungen werden durch ein spezielles Schlüsselwortmuster ausgelöst, das von Dollarzeichen eingeschlossen wird. Zum Beispiel wird

```
$Revision$
```

in einer Datei zu einem Ausdruck wie

```
$Revision: 1.5 $
```

und CVS wird bei jedem `Commit` einer neuen Revision diese Textstelle auf dem neuesten Stand halten.

6.1 Schlüsselwort-Expansion kontrollieren

Normalerweise expandiert CVS Schlüsselwörter, solange Sie ihm nicht befehlen, das nicht zu tun. Sie können die Schlüsselwortersetzung für eine Datei permanent verhindern, indem Sie die `-k`-Option angeben, wenn Sie die Datei dem Projekt hinzufügen, oder Sie können sie später deaktivieren, indem Sie den Befehl admin mit der Option `-k` aufrufen. Die `-k`-Option bietet mehrere verschiedene Methoden der Schlüsselwortkontrolle; üblicherweise werden Sie `o` oder `b` benötigen, wie zum Beispiel:

```
user@linux ~/ # cvs add -ko neue_datei.txt
```

Dieser Befehl fügte die Datei `neue_datei.txt` dem Projekt hinzu, wobei die Schlüsselwortersetzung abgeschaltet ist. Es setzt die Schlüsselwortersetzungs-Methode der Datei auf `o` fest, was so viel wie **keine Ersetzung** bedeutet. (Genau genommen steht das **o** für **old**3, was bedeutet, dass das Wort mit seinem alten Wert - also durch sich selbst - ersetzt wird, was effektiv in keiner Änderung resultiert. Ich bin sicher, diese Logik ergab zum damaligen Zeitpunkt für irgendjemanden einen Sinn.)

Die Schlüsselwortersetzungs-Methode für jede einzelne Datei wird im Archiv gespeichert. Doch kann zudem jede Arbeitskopie ebenfalls ihre eigene lokale Einstellung besitzen - bewirkt durch die `-k`-Optionen zu `checkout` und `update`. Man kann außerdem Ersetzungsmethoden mit der `-k`-Option zu `diff` lediglich für die Dauer eines einzigen Befehls zur Anwendung bringen.

Hier sind alle möglichen Methoden, angegeben mit vorne angefügter `-k`-Option (so, wie man sie auf der Kommandozeile schreiben würde). Jede dieser Optionen kann sowohl als standardmäßige als auch als lokale Schlüsselwortersetzungs-Methode für eine Datei verwendet werden:

* `-kkv`
  * Schlüsselwort und Wert; dies ist die normale Schlüsselwortexpansions-Methode, daher muss sie nicht explizit für neue Dateien mit angegeben werden. Sie könnten sie jedoch benötigen, um eine Datei mit einer anderen Methode zurückzusetzen.

* `-kkvl`
  * Wie `-kkv`, beinhaltet aber zusätzlich den Namen desjenigen, der die Revision gebunden hat, falls diese auf einem bestimmten Wert gehalten wird. (Siehe die `-l`-Option zum Befehl admin für mehr Informationen dazu.)

* `-kk`
  * Schreibt keine Werte in Schlüsselwortfelder, sondern benutzt lediglich den Namen des Schlüsselwortes. Mit dieser Option würden zum Beispiel:

```
$Revision: 1.5 $
```

und

```
$Revision$
```

beide **expandiert** (okay, eher **kontrahiert**) zu:

```
$Revision$
```

* `-ko`
  * Wiederverwendung des Wortes, das in der Datei gefunden wird (**o** für **old**, wie oben beschrieben), gerade so, wie es vor dem `commit` in der Arbeitskopie stand.

* `-kb`
  * Wie `-ko`, zusätzlich wird aber noch die sonst zwischen unterschiedlichen Plattformen übliche automatische Umwandlung von Zeilenendungen unterdrückt. Das **b** steht für **binär**; es ist die Methode, die für Binärdateien verwendet werden sollte.

* `-kv`
  * Ersetzt das Schlüsselwort durch seinen Wert, so wird zum Beispiel

```
$Revision$
```

umgewandelt zu:

```
1.5
```

Natürlich wird in diesem Fall nach der ersten keine weitere Ersetzung stattfinden, sodass diese Methode mit Bedacht verwendet werden sollte.

7 Liste der Schlüsselwörter

Hier nun all die mit Dollarzeichen abgeschlossenen Schlüssewörter, die CVS kennt. Es folgt eine Liste mit je einem Schlüsselwort, einer kurzen Beschreibung und einem Beispiel seiner expandierten Form.

* `$Author$`
  * Autor einer Änderung
  * **$Author: jrandom $**

* `$Date$`
  * Datum und Zeit einer Änderung in UTC (GMT)
  * **$Date: 1999/08/23 18:21:13 $**

* `$Header$`
  * Diverse Informationen, die von Nutzen sein könnten: Der volle Pfad zur RCS-Datei im Archiv, Revision, Datum (UTC), Autor, Status und eventuell Benutzer, die einen Lock auf die Datei halten. (Letzere sind selten; in folgendem Beispiel hält jedoch der User qsmith einen Lock.):
  * **$Header: /usr/local/archiv/proj/hallo.c,v 1.1 1999/06/01 03:21:13 \ jrandom Exp qsmith $**

* `$Id$`
  * Wie `$Header$`, aber ohne den vollen Pfad zur RCS-Datei:
  * **$Id: hallo.c,v 1.1 1999/06/01 03:21:13 jrandom Exp qmsith $**

* `$Log$`
  * Die Log-Mitteilung für diese Revision zusammen mit Revisionsnummer, Datum und Autor. Anders als bei anderen Schlüsselwörtern werden frühere Ersetzungen nicht ersetzt. Sie werden stattdessen nach unten verschoben, sodass die neueste Ersetzung immer ganz oben in einer ständig wachsenden Liste von `$Log$`-Nachrichten steht:
  * **$Log: hallo.c,v $ Revision 1.12 1999/07/19 06:12:43 jrandom**
  * Halli hallo, eine Mitteilung
  * Text, der vor dem `$Log$`-Schlüsselwort auf derselben Zeile steht, wird ebenfalls mit der $Log-Mitteilung nach unten verschoben; das geschieht, damit alles, was später expandiert wird, ebenfalls auskommentiert ist, wenn Sie `$Log$` in einem Kommentar in einer Quelltextdatei verwenden.

* `$Locker$`
  * Name des Benutzers, der einen Lock auf diese Revision hält (normalerweise niemand):
  * **$Locker: qsmith $**

* `$Name$`
  * Name der gebundenen Markierung:
  * **$Name: release_1_14 $**

* `$RCSfile$`
  * Name der RCS-Datei im Archiv:
  * **$RCSfile: hallo.c,v $**

* `$Revision$`
  * Revisionsnummer:
  * **$Revision: 1.1 $**

* `$Source$`
  * Vollständiger Pfad zur RCS-Datei im Archiv:
  * **$Source: /usr/local/archiv/proj/hallo.c,v $**

* `$State$`
  * Status dieser Revision:
  * **$State: Exp $**

8 Archivverwaltungsdateien

Die Verwaltungsdateien eines Archivs werden im `CVSROOT`-Unterverzeichnis des Archivs gespeichert. Diese Dateien kontrollieren diverse Aspekte im Verhalten von CVS (selbstverständlich nur innerhalb dieses Archivs).

Generell werden die Verwaltungsdateien genauso unter Revisionskontrolle gehalten wie alle anderen Dateien im Archiv (Ausnahmen werden extra angemerkt). Im Gegensatz zu anderen Dateien werden ausgecheckte Kopien der Verwaltungsdateien im Archiv jedoch direkt neben den korrespondierenden RCS-Dateien abgespeichert. Diese ausgecheckten Kopien üben die wirkliche Kontrolle über das Verhalten von CVS aus.

Der normale Weg, um die Verwaltungsdateien zu modifizieren, ist der, einen `Checkout` auf eine Arbeitskopie des CVSROOT-Moduls durchzuführen, die Änderungen vorzunehmen und einen `Commit` zu machen. CVS aktualisiert die ausgecheckten Dateien im Archiv automatisch (siehe `checkoutlist`). In Notfällen ist es jedoch auch möglich, die ausgecheckten Kopien im Archiv direkt zu bearbeiten.

Eventuell interessieren Sie sich auch für die Beschreibung von Verwaltungsdateien in Kapitel 4, die auch Beispiele umfasst.

8.1 Gemeinsame Syntax

In allen Verwaltungsdateien markiert ein # am Zeilenbeginn einen Kommentar; diese Zeile wird von CVS nicht beachtet. Ein Backslash \ vor einem Zeilenende bewirkt das Ignorieren des Zeilenumbruchs.

Einige der Dateien (`commitinfo`, `loginfo`, `taginfo` und `rcsinfo`) haben weitere syntaktische Konventionen gemeinsam. Innerhalb dieser Dateien befindet sich am linken Rand jeder Zeile ein regulärer Ausdruck (der mit einem Datei- oder Verzeichnisnamen verglichen wird), und der Rest der Zeile ist ein Programmaufruf, eventuell mit Argumenten, der ausgeführt wird, sobald irgendetwas mit der Datei geschieht, auf die der reguläre Ausdruck passt. Das Programm wird mit dem Hauptverzeichnis des Archivs als Arbeitsverzeichnis gestartet.

In diesen Dateien dürfen zwei spezielle reguläre Ausdrücke verwendet werden: ALL und DEFAULT. Der Ausdruck ALL passt für jeden beliebigen Datei- oder Verzeichnisnamen, egal ob noch andere Ausdrücke dafür passen oder nicht, und DEFAULT wird nur verwendet, falls keiner der anderen Ausdrücke passt.

8.2 Gemeinsame Variablen

Die Info-Dateien erlauben des Weiteren die Expansion gewisser Variablen zur Laufzeit. Um eine Variable zu expandieren, muss ihr ein Dollarzeichen vorangestellt sein (und setzen Sie sie in geschweifte Klammern, um sicherzugehen). Es folgen die Variablen, die CVS bekannt sind:

* `${CVSROOT}`
  * Das Hauptverzeichnis des Archivs

* `${RCSBIN} (Überholt)`
  * Benutzen Sie diese Variable nicht. Sie findet nur noch in alten CVS-Versionen (1.9.18 und älter) Verwendung.

* `${CVSEDITOR} ${VISUAL} ${EDITOR}`
  * Alle diese Variablen expandieren zu dem Editor, den CVS für Log-Mitteilungen verwendet.

* `{USER}`
  * Der Benutzer, der CVS laufen lässt (auf der Serverseite).

Benutzervariablen

Benutzer können zudem eigene Variablen setzen, wenn sie beliebige CVS-Kommandos starten (siehe die globale `-s`-Option). Auf diese Variablen kann aus den `*info`-Dateien zugegriffen werden, indem ihnen ein Gleichheitszeichen vorangestellt wird wie z.B. in `${=VAR}`.

8.3 Liste der Archivverwaltungsdateien

Es folgt eine Liste aller Archivverwaltungsdateien:

`checkoutlist`

Diese Datei enthält eine Liste von Dateien, von denen ausgecheckte Kopien im Archiv gehalten werden sollten. Jede Zeile enthält einen Dateinamen und eine Fehlermeldung, die CVS ausgeben soll, falls aus irgendeinem Grund die Datei nicht im Archiv ausgecheckt werden kann.

DATEINAME FEHLERMELDUNG

Da CVS bereits von sich aus ausgecheckte Kopien der existierenden Verwaltungsdateien behält, müssen diese nicht in der `checkoutlist`-Datei aufgeführt werden. Speziell die folgenden Dateien brauchen niemals in der `checkoutlist`-Datei zu stehen: `loginfo`, `rcsinfo`, `editinfo`, `verifymsg`, `commitinfo`, `taginfo`, `ignore`, `checkoutlist`, `cvswrappers`, `notify`, `modules`, `readers`, `writers` und `config`.

`commitinfo`

Spezifiziert Programme, die, in Abhängigkeit von den betroffenen Dateien, bei einem `Commit` gestartet werden sollen. Jede Zeile besteht aus einem regulären Ausdruck, gefolgt von einem Kommando:

`REGULÄRER_AUSDRUCK PROGRAMM [ARGUMENTE]`

Dem Kommando werden außer den in der Datei vermerkten Argumenten zusätzliche Parameter übergeben, und zwar der volle Pfad zum Archiv gefolgt von den Namen aller Dateien, die per `Commit` übertragen werden sollen. Diese Dateien können durch das PROGRAMM untersucht werden; ihr Inhalt entspricht dem der Dateien aus der Arbeitskopie, auf die `commit` angewendet werden soll. Beendet sich das PROGRAMM mit einem Rückgabewert ungleich null, so schlägt `commit` fehl; anderenfalls wird der `Commit` durchgeführt. (Siehe auch **Gemeinsame Syntax** weiter vorne in diesem Kapitel.)

`config`

Kontrolliert diverse globale (nicht projektspezifische) Archivparameter. Die Syntax jeder Zeile ist:

ParameterName=yes|no

mit Ausnahme des LockDir Parameters, der einen absoluten Pfadnamen als Argument benötigt.

Die folgenden Parameter werden unterstützt:

RCSBIN (normal: =no)

(Überholt.) Diese Option wird aus Kompatibilitätsgründen stillschweigend akzeptiert, hat jedoch keinerlei Wirkung mehr.

SystemAuth (normal: =no)

Wenn **yes**, dann konsultiert die pserver Authentifizierung die Benutzerdatenbank des Systems - normalerweise

`/etc/passwd` - falls ein Benutzer nicht in `CVSROOT/passwd` gefunden werden kann. Wenn **no**, muss der Benutzer in `CVSROOT/passwd` stehen, um Zugriff über die `:pserver:`-Methode zu erhalten.

PreservePermissions (normal: =no)

Wenn **yes**, dann versucht CVS Dateizugriffsrechte und andere spezielle Dateisysteminformationen (wie Device-Nummern und Ziele symbolischer Links) möglichst zu erhalten. Dies möchten Sie wahrscheinlich nicht, weil diese Option sich nicht unbedingt immer verhält wie erwartet. (Siehe auch den Abschnitt **Special Files** im *Cederqvist* mit weiteren Details.)

TopLevelAdmin (normal: =no)

Wenn **yes**, dann erzeugen `Checkouts` ein `CVS/`-Unterverzeichnis neben jedem Dateibaum der Arbeitskopie (im Mutterverzeichnis der Arbeitskopie). Dies kann nützlich sein, wenn Sie viele Arbeitskopien aus demselben Archiv auschecken; andererseits wird durch eine Einstellung an dieser Stelle jeder Benutzer des Archivs betroffen.

`LockDir` (normal: nicht gesetzt)

Das Argument nach dem Gleichheitszeichen ist der Pfad zu einem Verzeichnis, in dem CVS Lock-Dateien erzeugen kann. Wenn dies nicht gesetzt ist, werden Lock-Dateien innerhalb des Archivs jeweils bei den korrespondierenden RCS-Dateie jedes Projektes angelegt. Das bedeutet, dass Benutzern dieser Projekte auf Dateisystemebene Schreibzugriff auf diese Archivverzeichnisse gewährt sein muss.

`cvsignore`

Ignoriert bestimmte Dateien, wenn Aktualisierungen, `Imports` oder `Releases` durchgeführt werden. Normalerweise ignoriert CVS sowieso einige Arten von Dateien. (Für eine vollständige Liste siehe die `-I` Option zu import weiter vorne in diesem Kapitel.) Sie können diese Liste erweitern, indem Sie zusätzliche Dateinamen oder Wildcards in der `cvsignore`-Datei eintragen. Jede Zeile gibt einen Dateinamen oder ein Muster, z.B.:

```
README.msdos

*.html

blah?.out
```

Hierdurch wird CVS veranlasst, alle Dateien mit dem Namen **README.msdos**, alle Dateien, die auf **.html** enden, und alle Dateien, die mit **blah** anfangen und mit **.out** enden, zu ignorieren. (Theoretisch könnten Sie mehrere Dateien oder Wildcards in jeder Zeile durch Leerzeichen getrennt angeben, aber die Lesbarkeit ist bei einem Ausdruck pro Zeile besser. Bedauerlicherweise impliziert diese Tatsache auch, dass es keinen Weg gibt, Dateinamen, die Leerzeichen enthalten, zu spezifizieren, außer unter Verwendung von Wildcards.)

Ein **!** an beliebiger Stelle innerhalb der Liste macht alle vorangehenden Einträge unwirksam. (Siehe `$CVSIGNORE` im bschnitt **Umgebungsvariablen** in diesem Kapitel für eine vollständigere Diskussion des Verhaltens von Ignore.)

`cvswrappers`

Definiert gewisse Filtermethoden, basierend auf dem Dateinamen. Jede Zeile enthält einen Dateinamen oder eine Wildcard, gefolgt von einer Option, die den Filtertyp festlegt, und ein Argument zu dieser Option.

Optionen:

* `-m`
  * Spezifiziert eine Aktualisierungsmethode. Mögliche Argumente sind MERGE, was eine automatische Integration in die Arbeitskopie bewirkt, sowie COPY, was bewirkt, dass keine Integration versucht wird, sondern der Benutzer mit beiden Versionen einer Datei konfrontiert wird und das selber regeln muss.

Standardeinstellung ist MERGE, außer bei Binärdateien (solche, deren Schlüsselwortersetzungs-Methode auf `-kb` gesetzt ist). (Siehe auch den Abschnitt **Schüsselwortersetzung** weiter vorne in diesem Kapitel.) Dateien, die als binär markiert sind, nutzen automatisch die `COPY` Methode, sodass kein Grund besteht, für diese extra einen `-m COPY-Wrapper` zu definieren.

* `-k`
  * Legt eine Schlüsselwortersetzungs-Methode fest. Alle üblichen Methoden sind erlaubt. (Siehe auch den Abschnitt **Schüsselwortersetzung** weiter vorne in diesem Kapitel mit einer vollständigen Liste.)

Hier ist ein Beispiel für eine `cvswrappers`-Datei:

```
*.blob -m COPY

*.blink -k o
```

Diese `cvswrappers`-Datei legt fest, dass für Dateien, die auf `.blob` enden, keine automatische Integration versucht werden soll und für Dateien, die auf `.blink` enden, die Schlüsselwortersetzung unterdrückt wird. (Siehe auch `.cvswrappers` im Abschnitt **Dateien in der Arbeitskopie** in diesem Kapitel.)

`editinfo`

Diese Datei ist überholt. Sehr überholt.

`history`

Speichert eine ständig anwachsende Historie von Aktivitäten im Archiv, zur Benutzung durch das `cvs history`-Kommando. Um diese Funktion zu deaktivieren, löschen Sie einfach die `history`-Datei. Wenn sie existiert, sollte sie am besten für alle beschreibbar sein, um spätere Probleme mit Dateizugriffsrechten zu vermeiden.

Der Inhalt dieser Datei ändert in keiner Weise das Verhalten von CVS (selbstverständlich mit Ausnahme der Ausgabe des Befehles `cvs history`).

`loginfo`

Spezifiziert Programme, welche die Log-Mitteilungen bei jedem `Commit` bearbeiten, abhängig davon, was von dem `Commit` betroffen ist. Jede Zeile setzt sich aus einem regulären Ausdruck, gefolgt von einem Befehlsausdruck, zusammen:

```
REGULÄRER_AUSDRUCK PROGRAMM [ARGUMENTE]
```

Dem PROGRAMM wird die Log-Mitteilung auf die Standardeingabe übermittelt.

Einige spezielle Codes können in ARGUMENTE verwendet werden: `%s` symbolisiert die Namen der Dateien, die der `Commit` betrifft, `%V` steht für die alten Revisionen vor dem `Commit` und `%v` für die neuen Revisionen nach dem `Commit`. Wenn mehrere Dateien betroffen sind, ist jedes Element der Ersetzung durch Leerzeichen vom nächsten getrennt. Beispielsweise wird bei einem `Commit`, der zwei Dateien betrifft, `%s` zu `hallo.c README.txt` und `%v` zu 1.17 1.12.

Sie können die Codes in geschweiften Klammern kombinieren, wodurch jeder zu einem Namen gehörige Block intern durch Kommata, die Blöcke an sich durch Leerzeichen getrennt werden. Um das vorige Beispiel weiterzuführen, sähe beispielsweise die Expansion von `%{sv}` so aus: `hallo.c,1.17 README.txt,1.12`.

Wenn überhaupt irgendeine % Expansion durchgeführt wird, dann wird der Pfad im Archiv vorangestellt. Das heißt, das letzte Beispiel ergäbe in Wirklichkeit:

```
myproj hallo.c,1.17 README.txt,1.12
```

Wenn PROGRAMM mit einem Rückgabewert ungleich null beendet wird, so schlägt `commit` fehl; anderenfalls wird die Aktion fortgeführt. (Siehe auch den Abschnitt **gemeinsame Syntax** weiter vorn in diesem Kapitel.)

`modules`

Diese Datei bildet Namen auf Archivverzeichnisse ab. Der generelle Aufbau der Dateizeilen ist:

MODUL [OPTIONEN] [&ANDERES_MODUL...] [VERZEICHNIS] [DATEIEN]

VERZEICHNIS muss kein Top-Level-Projektverzeichnis sein - es kann genauso gut ein Unterverzeichnis sein. Werden irgendwelche DATEIEN angegeben, so besteht das Modul nur aus diesen DATEIEN in diesem VERZEICHNIS.

Ein Und-Zeichen, gefolgt von einem Modulnamen, bewirkt, dass die Expansion jenes Moduls auf dieser Zeile übernommen wird.

Optionen:

* `-a`
  * Dies ist ein Modul-**Alias**, was heißt, es expandiert buchstäblich zu allem, was nach den OPTIONEN folgt. In diesem Falle ist das VERZEICHNIS/DATEIEN-Verhalten deaktiviert, und alles nach den OPTIONEN wird als andere Module oder Archivverzeichnisse angesehen. Wenn Sie die `-a`-Option verwenden, können Sie bestimmte Verzeichnisse von anderen Modulen ausnehmen, indem Sie ihnen ein Ausrufezeichen (!) voranstellen. Zum Beispiel bedeutet

```
user@linux ~/ # top_proj -a !meinproj/a-verzeichnis !meinproj/b-verzeichnis
meinproj
```

* * dass ein `Checkout` von **top_proj** alle Dateien in meinproj betrifft, außer denen in den Verzeichnissen a-verzeichnis und b-verzeichnis.

* `-d NAME`
  * Nennt das Arbeitsverzeichnis NAME, anstatt den Modulnamen zu verwenden.

* `-e PROGRAMM`
  * Startet PROGRAMM, wenn Dateien in diesem Modul exportiert werden.

* `-i PROGRAMM`
  * Startet PROGRAMM, wenn `Commits` von Dateien in diesem Modul stattfinden. Dem Programm wird ein Argument übergeben - der vollständige Pfad zu der fraglichen Datei innerhalb des Archivs. (Siehe auch `commitinfo`, `loginfo` und `verifymsg` für ausgeklügeltere Methoden zum Starten von Programmen bei einem `Commit`.)

* `-o PROGRAMM`
  * Startet PROGRAMM, wenn Dateien in diesem Modul ausgecheckt werden.

* `-s STATUS`
  * Legt einen Status für dieses Modul fest. Wenn die modules-Datei ausgegeben wird (mit `cvs checkout -s`), dann werden die Module zunächst nach Status, dann nach Namen sortiert. Diese Option hat keine weitere

Wirkung auf CVS, also nutzen Sie sie nach Belieben. Sie können Sie benutzen, um nach beliebigen Kriterien zu sortieren: beispielsweise Status, Verantwortliche für den Quelltext oder Sprache der Datei.

* `-t PROGRAMM`
  * Startet PROGRAMM, wenn Dateien in diesem Modul mittels `cvs rtag` markiert werden. Dem Programm werden zwei Argumente mitgegeben: der Name des Moduls und der symbolische Name der Markierung. Das Programm wird nur für rtag aufgerufen, nicht für `tag`. Ich habe keine Ahnung, warum diese Unterscheidung gemacht wird. Die `taginfo` -Datei könnte nützlicher für Sie sein, wenn Sie Programme durch tag starten möchten.

* `-u PROGRAMM`
  * Startet PROGRAMM, immer wenn eine Arbeitskopie dieses Moduls aus seinem Hauptverzeichnis heraus aktualisiert wird. Dem Programm wird ein einzelnes Argument übergeben: Der vollständige Dateiname zum Archiv des Moduls.

`notify`

Kontrolliert, wie die Benachrichtigungen für beobachtete Dateien durchgeführt werden. (Möglicherweise möchten Sie hierzu auch bei den `watch`- und `edit`-Kommandos nachlesen oder den Abschnitt **Watches** in Kapitel 6 anschauen.) Jede Zeile hat die übliche Form:

```
REGULÄRER_AUSDRUCK PROGRAMM [ARGUMENTE]
```

Ein `%s` innerhalb der ARGUMENTE wird zu dem Namen des Benutzers expandiert, der benachrichtigt werden soll, während der Rest der Information bezüglich der Benachrichtigung dem PROGRAMM auf der Standardeingabe übergeben wird. Üblicherweise ist diese Information eine kurze Nachricht, die dazu geeignet ist, eine E-Mail an den Benutzer zu erzeugen. (Siehe auch den Abschnitt **gemeinsame Syntax** weiter vorne in diesem Kapitel.)

In der Standardkonfiguration von CVS hat die `notify`-Datei eine Zeile:

```
ALL mail %s -s "CVS notification"
```

die meistens auch vollständig ausreicht.

`passwd`

Enthält Authentisierungsdaten zur Nutzung mit der pserver-Zugriffsmethode. Jede Zeile hat die Form:

```
BENUTZER:VERSCHLÜSSELTES_PASSWORT[:SYSTEM_BENUTZER]
```

Ist kein SYSTEM_BENUTZER gegeben, wird BENUTZER als der Systembenutzername angenommen.

`rcsinfo`

Definiert ein Formular, das für Log-Mitteilungen ausgefüllt werden sollte, die mit einem interaktiven Editor erstellt werden. Jede Zeile von rcsinfo sieht wie folgt aus:

```
REGULÄRER_AUSDRUCK DATEI_MIT_FORMULARVORLAGE
```

Die Vorlage wird an extern gelagerte Arbeitskopien übergeben, wenn diese einen `Checkout` durchführen, sodass eventuelle Änderungen an der Vorlage zunächst ohne Wirkung bleiben, da die externen Kopien die alte Version weiterverwenden. (Siehe auch den Abschnitt **gemeinsame Syntax** in diesem Kapitel.)

`taginfo`

Startet ein Programm beim Setzen von Markierungen. (Üblicherweise um zu überprüfen, dass die Markierungen einem bestimmten Schema folgen). Jede Zeile hat die Form:

```
REGULÄRER_AUSDRUCK PROGRAMM
```

Dem Programm wird ein Satz von Argumenten übergeben. In dieser Reihenfolge sind das der symbolische Name der Markierung, die Operation (siehe unten), das Archiv und so viele Paare aus Dateiname/Revision, wie Dateien von der Aktion betroffen sind. Die Datei/Revision-Paare sind durch Leerzeichen getrennt, wie der Rest der Argumente.

Die Operation ist entweder add, mov oder del. (mov bedeutet, dass die `-F`-Option des `tag`-Befehls verwendet wurde.)

Wenn PROGRAMM mit einem Rückgabewert ungleich null beendet wird, so schlägt der `tag`-Befehl fehl; andernfalls wird er fortgeführt. (Siehe auch den Abschnitt **gemeinsame Syntax** weiter vorn in diesem Kapitel.)

`users`

Verknüpft Benutzernamen mit E-Mail-Adressen. Jede Zeile sieht aus wie:

```
BENUTZERNAME:EMAIL_ADRESSE
```

Hiermit können Benachrichtigungen über beobachtete Dateien an EMAIL_ ADRESSE anstatt an den BENUTZERNAMEN auf der Maschine mit dem Archiv versendet werden. (Alles, was hier gemacht wird, ist die Kontrolle der Expansion von `%s` in der `notify`-Datei.) Sollte die EMAIL_ADRESSE Leerzeichen enthalten, dann stellen Sie sicher, dass sie in Anführungszeichen steht.

Wenn Benutzer-Aliasing in der passwd-Datei verwendet wird, dann ist der Benutzername, der hier verglichen wird, der CVS-Benutzername (der auf der linken Seite), nicht der Systembenutzername (rechts, falls vorhanden).

`val-tags`

Diese Datei dient als Zwischenspeicher für gültige Markierungen, um deren Auffindung zu beschleunigen. Es sollte nie notwendig sein, dass Sie diese Datei bearbeiten müssen, aber es könnte notwendig sein, die Zugriffs- oder Eigentumsrechte zu ändern, wenn Leute Probleme beim Zugriff auf oder bei der Erzeugung von Markierungen haben.

`verifymsg`

Wird in Verbindung mit `rcsinfo` verwendet, um das Format von Log-Mitteilungen zu verifizieren. Jede Zeile hat die Form:

```
REGULÄRER_AUSDRUCK PROGRAMM [ARGUMENTE]
```

Der vollständige Pfad zur aktuellen Vorlage für Log-Mitteilungen (siehe auch **rcsinfo** weiter vorne in diesem Kapitel) wird hinter dem letzten, in der verifymsg-Datei angegebenen Argument angefügt. Wenn PROGRAMM mit einem Rückgabewert ungleich null beendet wird, so schlägt `commit` fehl; andernfalls wird die Aktion fortgeführt. (Siehe auch den Abschnitt **gemeinsame Syntax** weiter vorn in diesem Kapitel.)

9 Laufzeit-Kontrolldateien

Es gibt nur einige wenige Dateien auf der Client-Seite (Arbeitskopie), die das Verhalten von CVS beeinflussen können. In einigen Fällen entsprechen sie den Archivverwaltungsdateien; in anderen Fällen kontrollieren sie Verhaltensweisen, die nur für die Client-Seite von Bedeutung sind.

9.1 .cvsrc

Enthält Optionen, die Sie automatisch bei jedem Aufruf von CVS-Kommandos verwenden möchten. Das Format jeder Zeile ist

```
KOMMANDO OPTIONEN
```

wobei jedes KOMMANDO ein nicht abgekürzter CVS-Befehl ist, wie z.B. `checkout` oder `update` (und nicht `co` oder `up`). Die OPTIONEN sind jene, die Sie grundsätzlich anwenden möchten, wenn der Befehl verwendet wird. Hier eine übliche `.cvsrc`-Zeile:

9.2 update -d -p

um globale Optionen für jedes Kommando zu setzen, nehmen Sie einfach cvs als KOMMANDO.

9.3 .cvsignore

Definiert zusätzliche Muster zum Ignorieren von Dateien. (Siehe `cvsignore` im Abschnitt **Archivverwaltungsdateien** in diesem Kapitel zur Syntax.)

Sie können eine `.cvsignore`-Datei in Ihrem `Home`-Verzeichnis haben, die stets verwendet werden wird, wenn Sie CVS benutzen. Sie können zudem verzeichnisspezifische Dateien in jedem Projektverzeichnis einer Arbeitskopie haben. Letztere wirken dann nur auf das Verzeichnis, in dem sie abgelegt sind, und nicht in dessen Unterverzeichnissen. (Siehe auch `$CVSIGNORE` im Abschnitt **Umgebungsvariablen** in diesem Kapitel.)

9.4 .cvspass

Speichert Passwörter für jedes Archiv, auf das Sie mit der pserver-Methode zugreifen. Jede Zeile hat die Form:

```
ARCHIV LEICHT_VERSCHLÜSSELTES_PASSWORT
```

Im Grunde werden die Passwörter im Klartext gespeichert - lediglich eine leichte Verdrehung der Daten soll bewirken, dass nicht per Zufall (z.B. wenn root versehentlich den Dateiinhalt anzeigt) die Daten direkt sichtbar sind. Diese Maßnahme wird jedoch keine Person, die es ernsthaft darauf anlegt, davon abhalten, das Passwort zu entschlüsseln, falls sie Zugriff auf die Datei erhält.

Die Datei `.cvspass` ist nicht an einen Rechner gebunden. Sie können es von einer Maschine auf eine andere kopieren und haben dann auch dort all Ihre CVS-Passwörter verfügbar, ohne dort jemals `cvs login` aufrufen zu müssen. (Siehe auch die Kommandos `login` und `logout`.)

9.5 .cvswrappers

Hierbei handelt es sich um eine client-seitige Version der `cvswrappers`-Datei. (Siehe den Abschnitt **Archivverwaltungsdateien** in diesem Kapitel.) Eine `.cvswrappers`-Datei darf sich in Ihrem `Home`-Verzeichnis sowie in jedem Verzeichnis der Arbeitskopie befinden, genau wie bei `.cvsignore`.

10 Dateien in der Arbeitskopie

Die `CVS/`-Verwaltungsunterverzeichnisse in jeder Arbeitskopie enthalten einige der folgenden Dateien:

10.1 CVS/Base/ (Verzeichnis)

Sollten `Watches` (siehe Kapitel 6) aktiviert sein, speichert `cvs edit` die Originalversion der Datei in diesem Verzeichnis. So kann cvs `unedit` auch dann funktionieren, wenn der Server gerade nicht erreichbar ist.

10.2 CVS/Baserev

Liste der Revisionen jeder Datei in `Base/`. Jede Zeile sieht so aus:

```
DATEI/REVISION/ERWEITERUNG
```

ERWEITERUNG ist ein momentan nicht beachtetes Feld für, nun ja, zukünftige Erweiterungen eben.

10.3 CVS/Baserev.tmp

Die temporäre Datei zur vorhergehenden. (Siehe `CVS/Notify.tmp` oder `CVS/Entries.Backup` später in diesem Abschnitt zur Erklärung.)

10.4 CVS/Checkin.prog

Nimmt den Namen des Programmes auf, das zu der `-i`-Option in der Datei modules angegeben ist. (Siehe auch den Abschnitt **Archivverwaltungsdateien** in diesem Kapitel.)

10.5 CVS/Entries

Speichert die Revisionen für die Dateien in diesem Verzeichnis. Jede Zeile hat die Form:

```
[KÜRZEL]/DATEI/REVISION/DATUM/[SCHLÜSSELWORT_METHODE]/[GEBUNDENE_OPTION]
```

Falls KÜRZEL vorhanden ist, muss er `D` für Directory (Verzeichnis) sein (alles andere wird von CVS ignoriert, um spätere Erweiterungen der Funktionalität zu erlauben), und der Rest der Bestandteile auf der Zeile fehlt. Diese Datei ist immer vorhanden.

10.6 CVS/Entries.Backup

Das ist nur eine temporäre Datei. Wenn Sie irgendein Programm schreiben, das die Datei `Entries` modifizieren soll, dann lassen Sie es das Ergebnis zunächst in die Datei `Entries.Backup` schreiben, und sie danach automatisch in **Entries** umbenennen.

10.7 CVS/Entries.Log

Hierbei handelt es sich im Grunde um einen Patch, der `Entries` hinzugefügt wird, nachdem `Entries` gelesen wurde (ein Hack aus Effizienzgründen, um nicht wegen jeder Kleinigkeit die ganze `Entries`-Datei neu schreiben zu müssen). Das Format ist im Grunde dasselbe wie bei Entries, nur dass hier jede Zeile mit einem Kürzel beginnen muss: Ein **A** (`add`) bedeutet, dass diese Zeile dem Inhalt von Entries hinzugefügt werden muss; **R** (`remove`), dass sie dem Inhalt von Entries abgezogen werden soll. Andere Buchstaben werden ignoriert, um zukünftige Erweiterungen zu erlauben.

10.8 CVS/Entries.Static

Wenn diese Datei existiert, so wurde nur ein Teil des Verzeichnisses aus dem Archiv übertragen, und CVS wird keine weiteren Dateien in diesem Verzeichnis erzeugen. Dieser Zustand kann normalerweise durch ein `update -d` behoben werden.

10.9 CVS/Notify

Speichert Benachrichtigungen, die noch nicht an den Server geschickt wurden.

10.10 CVS/Notify.tmp

Die temporäre Datei zu `Notify`. Die übliche Vorgehensweise zur Modifikation von `Notify` ist `Notify.tmp` zu schreiben und dann in **Notify** umzubenennen.

10.11 CVS/Repository

Der Pfad zu dem projektspezifischen Unterverzeichnis im Archiv. Das kann eine absolute Angabe sein oder eine relativ zu dem Pfad, der in Root angegeben ist.

Diese Datei ist immer vorhanden.

10.12 CVS/Root

Dies ist das Archiv; d.h. der Wert der `CVSROOT`-Umgebungsvariablen oder des Arguments zu der globalen Option `-d`.

Diese Datei ist immer vorhanden.

10.13 CVS/Tag

Ist eine Markierung oder ein Datum an dieses Verzeichnis gebunden, so wird das in der ersten Zeile dieser Datei vermerkt. Das erste Zeichen ist ein einzelner Buchstabe, der den Typ der Bindung bestimmt: **T**, **N**, oder **D** bezeichnen, ob es sich um die Bezeichnung eines Zweiges, einen symbolischen Namen (Marke) oder ein Datum handelt. Der Rest der Zeile ist die Markierung respektive das Datum selbst.

10.14 CVS/Template

Enthält eine Vorlage für Log-Mitteilungen entsprechend der Vorgabe durch die `rcsinfo`-Datei. (Siehe den Abschnitt **Archivverwaltungsdateien** weiter vorn in diesem Kapitel.) Diese Datei ist nur für externe Arbeitskopien von Bedeutung; lokale Arbeitskopien auf demselben Rechner wie das Archiv lesen `rcsinfo` direkt aus dem Archiv.

10.15 CVS/Update.prog

Enthält den Namen des Programmes, das mittels der `-u`-Option in der Datei modules angegeben wurde. (Siehe den Abschnitt **Archivverwaltungsdateien** in diesem Kapitel.)

11 Umgebungsvariablen

Es folgt eine Liste aller Umgebungsvariablen, die CVS kennt.

11.1 $COMSPEC

Wird nur in OS/2 verwendet; sie enthält den Namen des Kommando-Interpreters. Standardwert ist: **CMD.EXE**.

11.2 $CVS_CLIENT_LOG

Wird zur Fehlerbeseitigung im Client/Server-Protokoll verwendet. Wenn Sie in diese Variable einen Dateinamen schreiben, bevor Sie CVS starten, so wird der gesamte eingehende Datenverkehr in Dateiname`.in`, der ausgehende in Dateiname`.out` abgespeichert.

11.3 $CVS_CLIENT_PORT

Wird für Kerberos-authentisierten Client/Server-Zugriff benötigt.

11.4 $CVSEDITOR

Enthält den Namen des Editors, der zum Erstellen von Log-Mitteilungen genutzt werden soll. Diese Variable hebt eventuelle Definitionen in `$EDITOR` oder `$VISUAL` auf.

11.5 $CVSIGNORE

Eine durch Leerzeichen getrennte Liste von Namen und Wildcards, die von CVS ignoriert werden sollen. (Siehe auch die `-I`-Option des `import`-Kommandos.) Der Inhalt dieser Variablen wird jeder Ignore-Liste eines Kommandos als Letztes angehängt. Letztere Liste wird in der folgenden Reihenfolge aufgebaut: `CVSROOT/cvsignore`, die Datei `.cvsignore` in Ihrem `Home`-Verzeichnis, die `$CVSIGNORE`-Umgebungsvariable, möglicherweise per `-I` angegebene Optionen und zuletzt der Inhalt der `.cvsignore`-Dateien in den jeweils bearbeiteten Unterverzeichnissen der Arbeitskopie. Ein **!** als Ignore-Muster an einem beliebigen Punkt dieser Liste löscht die gesamte Liste bis zu diesem Punkt.

11.6 SCVS_IGNORE_REMOTE_ROOT

Seit kurzem überholt.

11.7 SCVS_PASSFILE

Weist CVS an, eine andere Datei als `.cvspass` im `Home`-Verzeichnis für Authentisierungsdaten zu benutzen. (Siehe **.cvspass** im Abschnitt **Laufzeit-Kontrolldateien** in diesem Kapitel.)

11.8 $CVS_RCMD_PORT

Spezifiziert die Port-Nummer, auf welcher der rcmd-Daemon auf dem Server kontaktiert werden kann. (Diese Variable wird momentan von Unix CVS-Clients ignoriert.)

11.9 $CVSREAD

Setzt Dateien in der Arbeitskopie beim `Checkout` und `Update` nach Möglichkeit auf **nur lesbar**. Die Standardeinstellung ist **lesen und schreiben**. (Siehe auch die globale Option `-r`.)

11.10 $CVSROOT

Diese Variable enthält den Pfad zum Archiv und kann durch die globale Option `-d` oder durch das innerhalb einer Arbeitskopie gespeicherte Archiv aufgehoben werden. Dem Pfad kann gemäß folgender Syntax eine Zugriffsmethode, ein Benutzer- und ein Rechnername vorangestellt werden:

```
[[:METHODE:][[BENUTZER@]RECHNER]:]/ARCHIV_PFAD
```

Siehe auch die Beschreibung der globalen Option `-d` am Anfang dieses Kapitels für eine Liste möglicher Methoden.

11.11 $CVS_RSH

Enthält den Namen des externen Programmes, das zur Verbindung mit dem Server für die Zugriffsmethode `:ext:` verwendet wird. Normaler Wert ist `rsh`, aber `ssh` ist eine übliche Alternative dazu.

11.12 $CVS_SERVER

Enthält den Namen des Programms zum Aufruf von CVS auf der Serverseite. Ist natürlich standardmäßig auf cvs gesetzt.

11.13 $CVS_SERVER_SLEEP

Verzögert den Start des Serverprozesses beim Aufruf um die in dieser Variablen angegebene Zahl an Sekunden. Wird nur bei der Fehlersuche benötigt, damit der Debugger Zeit hat, die Kontrolle zu übernehmen.

11.14 $CVSUMASK

Enthält die Standardeinstellung für Schreib-/Lesezugriffe auf Dateien im Archiv. (Am besten gar nicht benutzen - es funktioniert sowieso nicht mit Client/Server-CVS.)

11.15 $CVSWRAPPERS

Eine durch Leerzeichen separierte Liste von Dateinamen, Wildcards und Argumenten, die CVS für so genannte Wrapper verwenden soll. (Siehe `cvswrappers` im Abschnitt **Archivverwaltungsdateien** in diesem Kapitel für mehr Informationen).

11.16 $EDITOR

(siehe `$CVSEDITOR`)

11.17 $HOME $HOMEDRIVE $HOMEPATH

Der Ort, wo Dateien wie `.cvsrc`, `.cvspass` gefunden werden können (unter Unix wird nur `HOME` beachtet). In Windows NT sollten `HOMEDRIVE` und `HOMEPATH` gesetzt sein, unter *Windows 95* kann es sein, dass sie noch von Hand eingestellt werden müssen.

Bemerkung:

> In Windows 95 müssen Sie evtl. auch HOME setzen. Stellen Sie dabei sicher, dass der Pfad nicht in einem

Schrägstrich endet; benutzen Sie set HOME=C: oder einen ähnlichen Ausdruck.

11.18 $PATH

Überholt.

11.19 $TEMP $TMP $TMPDIR

Der Ort, an dem temporäre Dateien gespeichert werden sollen. (Der Server benutzt TMPDIR; Windows NT benutzt TMP). Das Setzen dieser Variablen auf der Client-Seite beeinflusst den Server nicht. Generell beeinflusst das Setzen dieser Variablen nicht den Ort, an dem temporäre Locks abgespeichert werden. (Siehe auch den Abschnitt **config** im Abschnitt **Archivverwaltungsdateien** in diesem Kapitel für mehr Informationen.)

11.20 $VISUAL

siehe **&$CVSEDITOR**

1. Anm. d. Übers.: Die korrekte Zeitzone für Deutschland ist CET bzw. (im Sommer) CEST

2. Anm. d. Übers.: tot

3. Anm. d. Übers.: alt