UNIX - Werkzeuge

1. SCCS, RCS, CVS, SVN und GIT - Versionsverwaltungssysteme

Wie verwaltet man komplexe Quelltexte ohne daß man mal vor dem Nichts steht???

Versionsverwaltungssysteme - Leistungen und Ziele

Was wird verwaltet?

Gewöhnliche Dateien - Text-Files, in der Regel keine Binärdaten
z.B.: Quelltexte, Texte, Makefiles, Konfigurationsfiles,...

Mehrere Versionen werden aufbegehoben und bei Bedarf wieder erzeugt.

Ziele von Versionsverwaltungssystemen:

- Kontrollierten Zugriff ermöglichen
- Frühere Versionen wiederherstellen
- Kontinuierliche Generationsfolge sichern
- Dokumentation der Änderungen
- Parallelen Zugriff organisieren - Synchronisation
- Speicherplatz sparen
- Kosten reduzieren
- Zeit sparen

Historisches:

SCCS – Source Code Control System

1972 von Marc J. Rochkim in den Bell–Laboratorien entwickelt
1973 in die UNIX–System eingefügt
kostenpflichtige Software, keine frei verfügbaren Quellen
beruht auf 13 Einzelkommandos

RCS – Revision Control System

1983 von Walter F. Tichy an der Purdue Universität in West Lafayette entwickelt
mehrer Versionen mit signifikanten Unterschieden verfügbar
frei verfügbare Quellen
beruht auf 10 Einzelkommandos

CVS – Concurrent Versions System

1986 erster Entwurf von Dick Grune (Ableitung von RCS)
1989 Programmierung von CVS durch Brian Berliner
CVS ist frei verfügbar, beruht auf einem Einzelkommando mit vielen Optionen.
Netzwerkfähig.

Karl Fogel, Moshe Bar: Open Source–Projekte mit CVS
MITP–Verlag Bonn 2002, ISBN 3–8266–0816–x
Gregor N. Purdy: CVS kurz & gut
O'Reilly 2001, ISBN 3–89721–229–3

SVN – Subversion (Next Generation Open Source Version Control)

Open Source Versions Control System
Erste Ideen 2000. Abgeleitet von CVS. Beseitigt die Unzulänglichkeiten von CVS:
- Direktory Versionierung
- bessere Fileverwaltung (History)
- Metadatenverwaltung
- Verbesserter Netzwerkzugriffe
  (Apache, SSH, separater Server)
- gut strukturiert
- gut verwaltbar
- sicherer als CVS.

Besteht aus einem Nutzerkommando und mehreren Administrations-kommandos, Anbindung an Eclipse

Ben Collins–Sussman, Brian W. Fitzpatrick, C. Michael Pilato
Version Control with Subversion
O'Reilly 2004, ISBN 0–596–00448–6

GIT – the fast version control system

```
Ablösung von BitKeeper
Ursprünglicher Enwurf von Linux Torvalds zur Enwicklung des Kernels

für sehr große, komplexe Quelltextbäume
           z.B.   Linux Kernel,
Nicht-lineare, verteilte Entwicklung (branching und merging)
kein zentraler Server
lokale Repositories
Datentransfer zwischen Repositories
Kryptographische Sicherheit der Projektgeschichte
Gelöschte Daten bleiben vorhanden
Verbindungen zu CVS , SVN, ARCH
```


Übersicht – Was gibt es heute?

```
  Zentrale Versionsverwaltungssysteme:

      Open Source
         CVS
         SVN

      Propitäre Systeme
         Alienbrain
         Perforce
         Team Foundation Server
         Visual SourceSafe

  Verteilte Versionsverwaltungssysteme:

      Open Source:
         GNUarch  viele Kommandos, atomare Commits
         Bazaar   Abspaltung von arch (Ubuntu, MySQL, Inkscape, Emacs)
         Darcs    basiert auf Patchen, kein Baum von Revisionen
         GIT      (Linux-Kernel, ...)
         Mercurial (Firefox, Thunderbird, Google, OpenSolaris,
                    Linux-Kernel inoffiziell)
         Monotone  Ideenspender für Git (Pidgin)

      Propitäre Systeme
         BitKeeper (Linux-Kernel - alt - bis 2005)
         ClearCase
```

Arbeitsweise von Versionsverwaltungssystemen:

Versionsverwaltungssysteme benutzen in der Regel ein Repository als Speicher für die Daten. Der Nutzer agiert grundsätzlich in einer Sandbox, die eine Kopie des Repositories darstellt. Die Verwaltungssysteme organisieren den Datentransport zwischen Repository und Sandbox.

Initialisierung einer Datenbasis
    init

Füllen der Datenbasis mit einem Anfangszustand
    import

Ausgeben von Daten der Datenbasis an die Nutzer (Programmierer)
    checkout

Einfügen von Daten der Nutzer in die Datenbasis
    commit

Bilden einer neuen Version
    import

Aktualisieren der Datenbasis eines Nutzers
    update


```
CVS -  Concurrent Versions System
=================================

CVS wird mit kleinen Man-Pages ausgeliefert.

Mehr oder aktuellere Hilfen erhält man durch:
          info cvs      (GNU-info System)
          cvs --H  oder cvs --help
          cvs --help-options
          cvs --help-commands
          cvs --help <subkommando-name>

cvs --help
----------

Usage: cvs [cvs-options] command [command-options-and-arguments]
  where cvs-options are -q, -n, etc.
    (specify --help-options for a list of options)
  where command is add, admin, etc.
    (specify --help-commands for a list of commands
     or --help-synonyms for a list of command synonyms)
  where command-options-and-arguments depend on the specific command
    (specify -H followed by a command name for command-specific help)
  Specify --help to receive this message

The Concurrent Versions System (CVS) is a tool for version control.
For CVS updates and additional information, see
    the CVS home page at http://www.cvshome.org/ or
    Pascal Molli's CVS site at http://www.loria.fr/~molli/cvs-index.html
```

```
cvs --help-options
------------------

CVS global options (specified before the command name) are:
    -H           Displays usage information for command.
    -Q           Cause CVS to be really quiet.
    -q           Cause CVS to be somewhat quiet.
    -r           Make checked-out files read-only.
    -w           Make checked-out files read-write (default).
    -n           Do not execute anything that will change the disk.
    -t           Show trace of program execution (repeat for more
                 verbosity) -- try with -n.
    -R           Assume repository is read-only, such as CDROM
    -v           CVS version and copyright.
    -T tmpdir    Use 'tmpdir' for temporary files.
    -e editor    Use 'editor' for editing log information.
    -d CVS_root  Overrides $CVSROOT as the root of the CVS tree.
    -f           Do not use the ~/.cvsrc file.
    -z #         Use compression level '#' for net traffic.
    -a           Authenticate all net traffic.
    -s VAR=VAL   Set CVS user variable.
(Specify the --help option for a list of other help options)
```



```
cvs --help-commands
-------------------

CVS commands are:
        add          Add a new file/directory to the repository
        admin        Administration front end for rcs
        annotate     Show last revision where each line was modified
        checkout     Checkout sources for editing
        commit       Check files into the repository
        diff         Show differences between revisions
        edit         Get ready to edit a watched file
        editors      See who is editing a watched file
        export       Export sources from CVS, similar to checkout
        history      Show repository access history
        import       Import sources into CVS, using vendor branches
        init         Create a CVS repository if it doesn't exist
        log          Print out history information for files
        login        Prompt for password for authenticating server
        logout       Removes entry in .cvspass for remote repository
        ls           List files available from CVS
        pserver      Password server mode
        rannotate    Show last revision where each line of module was modifie
        rdiff        Create 'patch' format diffs between releases
        release      Indicate that a Module is no longer in use
        remove       Remove an entry from the repository
        rlog         Print out history information for a module
        rls          List files in a module
        rtag         Add a symbolic tag to a module
```

```
        server       Server mode
        status       Display status information on checked out files
        tag          Add a symbolic tag to checked out version of files
        unedit       Undo an edit command
        update       Bring work tree in sync with repository
        version      Show current CVS version(s)
        watch        Set watches
        watchers     See who is watching a file
(Specify the --help option for a list of other help options)
```



```
cvs -H import
-------------

Usage: cvs import [-d] [-k subst] [-I ign] [-m msg] [-b branch]
    [-W spec] repository vendor-tag release-tags...
        -d      Use the file's modification time as the time of import.
        -k sub  Set default RCS keyword substitution mode.
        -I ign  More files to ignore (! to reset).
        -b bra  Vendor branch id.
        -m msg  Log message.
        -W spec Wrappers specification line.
(Specify the --help global option for a list of other help options)

cvs --version
-------------

Concurrent Versions System (CVS) 1.12.12 (client/server)

Copyright (C) 2005 Free Software Foundation, Inc.

Senior active maintainers include Larry Jones, Derek R. Price,
and Mark D. Baushke.  Please see the AUTHORS and README files from the CVS
distribution kit for a complete list of contributors and copyrights.

CVS may be copied only under the terms of the GNU General Public License,
a copy of which can be found with the CVS distribution kit.

Specify the --help option for further information about CVS
```

Arbeitsweise von CVS

CVS führt die Informationen über den Werdegang eines Projektes mit. Alle Informationen werden in einem Repository gespeichert. Dieses enthält alle Informationen, die für die Wiedererstellung einer beliebigen Version notwendig sind. Das Repository wird durch einen Administrator verwaltet.

Der Nutzer arbeitet in einer oder mehreren Sandboxen (Sandkasten), die aktuelle Kopien von Files aus dem Repository enthalten (einer bestimmten Version).
Normalerweise holt sich der Nutzer ein zu bearbeitendes File aus dem Repository in seine Sandbox, bearbeitet es dort (testen der Änderung) und gibt es dann an das Repository zurück.

Prinzipiell sperrt CVS eine ausgecheckte Datei standardmäßig nicht, d.h. es können zwei Nutzer gleichzeitig an der selben Datei arbeiten. Der Konfikt wird erst bei der Rückgabe bemerkt. Die zweite Rückgabe wird unterdrückt. und der zweite Nutzer muß anschließend seine Änderung neu in die Datei einbringen. Der Administrator kann aber in Außnahmefällen einen Sperrmechnismus aktivieren (cvs admin -l ... oder cvs admin -L ...).

Ein einführendes Beispiel

Anlegen eines Nutzers cvs mit:

```
  Homedirectory:  /home/cvs  und Gruppe: cvsuser
```

CVS-User sollten zu cvsuser gehören.

Anlegen des Directories für ein Repository

```
 z.B. /usr/local/CVS
     Eigentümer:  cvs
     Gruppe:   cvsuser
     chmod g+ws /usr/local/CVS
```

Festlegen des Repository

```
   in   initcvs:   export CVS_RSH=ssh
                   export CVSROOT=bell@localhost:/usr/local/CVS
```

Setzen der Umgebungsvariablen:

```
   . initcvs
```

Initialisieren von cvs nur einmal durch den Administrator!!!!!!:

```
   cvs init
```

```
Initialisieren des Datenbestandes im Repository:

   cd ~/Tools/CVS
   ls -lisa
   drwx-------- 4 bell unixsoft 1157 Jan 29 14:02  Einleitung
   cd Einleitung
   cvs import -m 'Einleitung-Beispiel' Einleitung Bell V1_0

          hierduch wurde ein Projekt: Einleitung
              mit dem Hersteller: Bell
              und der Version:  V1_0
          mit den Files aus dem aktuellen Dirctory angelegt.

Erstes aus-checken aller Daten in die lokale Sandbox

   cd ~/Work     # Sandbox
   cvs checkout Einleitung   # Projekt: Einleitung wird geholt

Bearbeiten eines Files

   cvs checkout  Einleitung/sysconf.c
                    oder
   cvs update Einleitung
   cd Einleitung
   vi sysconf.c
   cvs commit sysconf.c
   ...
```



```
Kommandoaufbau:
---------------
 cvs {<Globale Optionen>} <CVS Subkommandos> \
              {<Optionen für Subkommandos> {<Bezeichner>}

Globale Optionen von CVS

  Folgende Optionen müssen vor dem Kommandoname spezifiziert werden:

    -H            Hilfsinformationen
    -Q            sehr schweigsames CVS
    -q            schweigsames CVS
    -r            mache checked-out Files read-only
    -w            mache checked-out Files read-write (standard)
    -l            Befehl nicht im History-Mechanismus
    -m infotext   Beschreibung
    -n            mache nichts, was die Daten verändert
    -t            trace -- bitte mit -n
    -v            CVS Versionsnummer und Copyright ausgeben
    -T tmpdir     Benutze 'tmpdir' für temporäre Files
    -e editor     Benute 'editor' zur  Editieren der Log-Information
    -d CVS_root   überschreibe $CVSROOT als root des CVS-Baums
    -f            benutze nicht das ~/.cvsrc File
    -z #          benutze Compressionsstufe '#' für den Netzwerkverkehr
    -a            Authentifzieren (pserver)
    -x            Verschlüsseln
    -s VAR=VAL    Setzen von CVS-Nutzervariablen
   --help         Hilfstexte
   --help-options Hilfstexte
   --help-command Hilfstexte
```

```
CVS Client-Subkommandos
-----------------------
  allgemeine Optionen für Client-Subkommandos
    (stehen nach dem CVS Client-Subkommando)

   -D datum  - die aktuellste Version vor dem Datum <datum> benutzen
               <datum>:  11-Nov-04, 1 month ago, last year, last Monday
                         yesterday, 3/31/92 10:00:07 PST, 22:00 GMT

   -f        - Befehl für Dateien anwenden, die nicht zu einer Markierung
               passen.

   -k kflag  - Schlüsselwortersetzungen - zum Einfügen von Versionsinformatio
                 kflag:  b    - binär
                         k    - nur Schlüsselwort
                         kv   - Schlüsselwortersetzung, Schlüsselwort wird
                                durch den zugehörigen Wert ersetzt (default)
                         kvl  - wie kv, aber Nutzerkennung wird hinzugefügt,
                         o    - Schlüsselwerte aus dem Repository  benutzen
                         v    - Nur Wert benutzen.

   -l        - Ausführung nicht rekursive in Unterverzeichnissen fortsetzen
               nur lokales Direktory

   -n        - keine Module-Programme ausführen

   -R        - rekursiv

   -r rev    - benutze die Revisionsnummer 'rev'(revision oder
               symbolic names (tag) je nach Kommando)
```



```
  CVS-Subkommandos

     import [-b zweig] [-d] [-I muster] [-k kfalg] [-m infotext]
            [-W name] modul hersteller version

          Importieren eines kompletten Verzeichnisses als neuer Modul in das
          Repository. Neuer Kode, der bisher nicht überprüft wurde kann in
          das Repository eingegliedert werden.

            -b zweig    - Importieren eines Herstellerzweiges.
            -d          - Verwende Zeitstempel der Dateiänderung der Dateien
                          anstelle der aktuellen Systemzeit.
            -I muster   - Erkennungsmuster für zu ignorierende Dateien.
            -m infotext - Information infotext für Protokolleintrag.
            -W name     - Name des CVS-Wrappers.
            modul       - Name des neuen Moduls im Repository.
            hersteller  - Herstellerkennung.
            version     - Versionshinweis.

     z.B.
          cvs import -m 'Einleitung-Beispiel' Einleitung Bell V1_0
          cvs import -m 'Einleitungs-Beispiel' Einleitungs-Projekt Bell V1_0
```

```
checkout [-A] [-c|s] [-d dir [-N]] [ [ -D datum | -r rev ] -f ]
         [-j rev1 [-j rev2]] [-k kflag] [-l|-R] [-n] [-p] [-P]
         [modul ...]

     Kopieren der Dateien aus dem Repository in eine Sandbox
     im aktuellen Verzeichnis, wenn nicht anders spezifiziert.

        -A     - Setze Datum und Markierungen zurück
        -c     - Ausgabe auf stdout
        -d dir - Ersetzen des Standardmäßigen Directorynamens
                 durch 'dir'
        -j rev - Führt Verzweigungen zusammen, neue 'rev'
        -N     - Keine Verkürzung von Modulpfaden
        -p     - Ausgabe auf stdout mit RCS-Headern
        -P     - Leere Verzeichnisse löschen
        -s     - Statusanzeige
        -D Datum, -r rev, -f, -k kflag, -l, -R  -  Siehe oben.

    z.B.

       cvs checkout   Einleitung
       cvs checkout   Einleitung/sysconf.c
```



```
annotate   [ [ -D datum | -r rev ] -f ] [-l | -R ] datei ...

     Anzeigen aller Zeilen der spezifizierten Objekte mit Änderungs-
     datum und Nutzer.

     Optionen siehe oben.

      z.B.
         cvs annotate  sysconf.c


commit [-f|[-l|-R]] [-F datei | -m infotext ] [-n] [-r rev ] [datei ...]

     Übergabe einer Änderung aus der Sandbox an das Repository.

        -f          - Erzwingen der Übergabe auch ohne Änderung.
        -F datei    - Infotext aus der Datei  'datei' benutzen.
        -m infotext - benutze angegebenen Info-Text infotext
        -l, -R, -r rev  - Siehe oben.

     z.B.

        cvs  commit   sysconf.c
```

```
update [-A] [-d] [-D datum | -r rev] [-f] [-I muster] [-j rev1 [-j rev2]
       [-k kflag] [-l |-R] [-p] [-P] [-W name] [datei ...]

    Syncronisieren der Sandbox mit dem Repository., wenn
    die Files in der Sandbox gegenüber dem Repository verändert
    wurden und ein commit fehlgeschlagen ist. Eventuell werden
    Update-Informationen in die Files eingefügt.

      -A
      -I muster
      -j rev1
      -j rev2
      -p
      -P
      -W name

     z.B.

             cvs update sysconf.c
```



```
add [ -k kflag ] [ -m infotext ] datei | directory

     Vorbereitung des Hinzufügens von neuen Files 'datei' oder
     Direktories  'directory' zum Repository. Die Eintragung
     wird in der aktuellen Sandbox gemacht.  Durch ein späteres
     commit wird die Aktion ausgelöst.

     -m infotext  - Benutze angegebenen Info-Text 'infotext'.
     -k kflag     - Siehe oben.

      z.B.
           cvs add README
           cvs commit README

diff [-k kflag] [-l|-R] [diff-option] [[-r rev1 | -D datum1]
       [-r rev2 | -D datum2]] [datei  ...]

     Anzeigen der Differenz zwischen Versionen von Dateien
     diff-option gibt die Art des Vergleiches und die Ausgabeform an:
       --binary, --brief, -c, -C zeilen, --context==nzeilen, -t,
       --expand-tabs, -w, --ignore-all-space,-B,--ignore-blank-lines,
       -i, --ignore-case, -T, --initial-tab, -d, --minimal, -N,
       --new-file, -n, --rcs, -s, --report-identical-files, -p,
       --schow-c-function, -y, --side-by-side, -a, --text, -u,
       -U nzeilen, --unified=nzeilen, -V typ

     -k kflag, -l, -R, -D datum   -siehe oben
```

edit [-a aktion] [-l | -R] [datei ...]

Einleitung einer Dateibearbeitung. Zusammen mit watch. Für die Datei wird der Schreibzugriff erlaubt und eine Benachrichtigung an andere Benutzer, die watch benutzt haben, wird versendet. edit wird durch unedit oder commit wieder aufgelöst.

```
-a aktion  - Aktion spezifizieren, über die man informiert werden
             möchte. Möglich sind: edit, unedit, commit, all oder n
-l, -R siehe oben
```

unedit [-l | -R] [datei ...]

Beenden einer Dateibearbeitung für die angegebenen Dateien. Dateien sind wieder schreibgeschützt. Beobachter werden benachrichtigt.

editors [-l | -R] [datei ....]

Zeigt alle gegenwärtigen Bearbeiter (Benutzer von edit) der spezifizierten Objekte an.

```
-l , -R   - siehe oben
```


watch {{ on | off } | { add | remove } [-a aktion]}
[-l | -R] datei ....

Setzen von Beobachtungspunkten für Dateiveränderungen.

watchers [-l | -R] [datei ...]

Anzeigen von Nutzern, die Dateiverändrungen überwachen.

export [-d dir [-N]] [-D datum | -r rev] [-f] [-k kflag]
[-l|-R] [-n] modul ...

Kopieren der Files aus dem Repository, es wird aber keine Sandbox erzeugt, d.h. es entstehen keine CVS-Verzeichnisse.

```
-d dir  - Benutze Verzeichnisname 'dir' anstelle des
          Modulnamens für den Export.
-N      - Pfade nicht verkürzen.
-n      - Kein module-Programm ausführen.
```

```
history [-a|-u benutzer] [-b string] [-c] [-D datum] [-e| -x kennung]
        [-f datei | -m modul | -n modul | -p repository ]
        [-l] [-o] [-r rev] [-t markierung] [-T] [-w] [-z zone]
             [datei..]

     Anzeigen von diversen Informationen zur History.

       -a             - History für alle Benutzer wird angezeigt.
       -u benutzer    - History wird nur für den Benutzer 'benutzer'
                        angezeigt.
       -b string      - Zeigt die History rückwärts an, bis der
                        String 'string' in Modulname, Pfadname oder
                        Repositorypfad vorkommt.
       -c             - Zeigt jedes commit-Kommando an.
       -e             - Zeigt alles an.
       -x Kennung     - Zeigt History zu bestimmten Aktivitäten an
                        Folgende Kennungen werden durch update erzeugt:
                            C - Zusammenführungen mit manuellem Eingriff
                            G - Automatische Zusammenführungen
                            U - Arbeitskopie aus dem Repository kopiert
                            W - Arbeitskopie gelöscht
                        Folgende Kennungen werden durch commit erzeugt:
                            A - Zum ersten Mal hinzugefügt
                            M - Modifiziert
                            R - Gestrichen
       -f datei       - Zeigt die letze Aktion für die Datei datei an.
       -m modul       - Zeigt einen vollständigen Bericht für den
                        Modul  'modul'   an.
```



```
       -p repository - Zeigt die History für das Repository repository
                        an.
       -o             - Zeigt einen Bericht der z.Z. ausgecheckten Module
                        an.
       -t markierung - Zeigt einen Bericht seit dem Zeitpunkt an,
                        seitdem die Markierung  'markierung'  eingefügt
                        wurde.
       -T             - Zeigt alle Markierungen an.
       -w             - History nur für das aktulle Verzeichnis anzeigen.
       -z zone        - Zeigt die Zeitangaben bezogen auf die
                        angegebene Zeitzone   'zone'  an.
       -D datum, -l, -r rev  - siehe oben

    Kennungen im Protokoll:

       C   - Zusammenführungen mit Konflikten (Eingriff des
             Nutzers war erforderlich)
       G   - Zusammenführung ohne Konflikt
       U   - Arbeitskopie aus dem Repository kopiert
       W   - Arbeitskopie gelöscht
       A   - Text zum ersten Mal ins Repository kopiert
       M   - Text modifiziert
       R   - Text aus dem Repository gestrichen
       E   - Repository exportiert (export)
       F   - release-Operation
       O   - checkout des Repositories
       T   - rtag (Tag angelegt)
```

```
log [-b] [-d datum] [-h] [N] [-rversion] [-R] [-s status]
    [-t] [-wlogins] [datei ....]
```

Anzeigen von History-Informationen:

```
 -b          - Anzeigen der Version des Hauptzweiges.
 -d datum    - Datumsbereich ( d1<d2, d1<=d2, <=d, <d, d )
 -h          - Gibt nur Kopfzeilen aus.
 -N          - Gibt nur Markierungen aus.
 -rversion   - Gibt Protokoll über die Version aus.
 -s status   - Gibt Protokoll nur zum Spezifizierten Status aus.
 -t          - Gibt nur Kopfzeilen und beschreibenden Text aus.
 -wlogin     - Gibt Protokoll zu den Einfügungen des Nutzers 'login'
               aus.
```

login

Login in Server (pserver)

logout

Logout beim Server (pserver)


```
rdiff [-c| -s| -u]
      [{{-D datum1 | -r rev1 } [-D datum2 | -r rev2] } | -t]
      [-f] [-l | -R]   datei ...
```

Erzeugen von 'patch'-Format-Informationen zwischen Releasen. Mit den Patchanweisungen kann eine Version in die andere Version überführt werden.

```
 -c   -  Benutzung des Context-Formates von diff.
 -s   -  Übersicht über alle gänderten Dateien.
 -t   -  Zeigt Unterschiede zwischen den beiden aktuellsten
         Versionen.
 -u   -  Benutzung des Univied-Formates von diff.
```

release [-d] verzeichnis ...

Löschen einer Sandbox oder eines Teils der Sandbox.

```
 -d   - Veränderte Eintragungen werden nur gelöscht, wenn sie
        an das Repository zuvor übergeben wurden.
```

```
remove [-f ] [-l | -R] [datei ....]

    Löschen eines Files vom Repository. Die Datei wird nicht wirklich
    gelöscht. Sie kann mit add  wieder hergestellt werden.


rtag [-a] [-b] [-d] [-D datum] [-r rev] [-f] [-F]
     [-l | -R] [-n]  markierung modul ...

    Versieht eine bestimmte Version einer Gruppe von Datei mit der
    Markierung markierung. Wenn eine Datei vorher schon mit einer
    anderen Markierunge versehen war, wird diese nicht markiert.

     -a  - löscht Tags von "removed" Files
     -b  - Setzen einer Markierung
     -d  - Löscht eine Markierung
     -F  - "force", erzwingt eine Markierung
     -n  - keine Ausführung von "tag programm"


status [-l | -R] [-v] [datei ...]

    Anzeigen von Statusinformation für Files
      -v    - verbose
```



```
tag [-b]  [-c]  [-d]  [-D datum | -r rev]  [-f]  [-F]  [-l | -R]
      markierung datei

    Markieren der Dateien in der Sandbox:
     -b  - Erzeugt eine Markierung für einen Zweig.
     -c  - Prüft vor dem Markieren, daß die lokalen Dateien nicht
           geändert wurden.
     -d  - Löscht die angegebene Markierung.
     -f  - Benutzt die übergeordnete Version für die Markierung, wenn
           die angebebenen nicht existiert.
     -F  - force, Erzwingt die Markierung.
```

```
Von CVS ausgewertete Umgebungsvariable
--------------------------------------

CVSEDITOR         - von CVS benutzter Editor
EDITOR
VISUAL
CVSROOT           - Wurzelverzeichnis für Repositories auf dem Server
                        [<nutzer>@<host>:]<absoluter Pfad>
USER              - Nutzername für ein entferntes Repository
CVS_CLIENT_LOG    - Dateiname für Fehlerprotokoll bei Client-Server-Verbindunge
CVS_CLIENT_PORT   - Client-Prot für Serververbindungen
CVS_PASSFILE      - Passwortdatei
CVS_RCMD_PORT     - reserviert nicht UNIX-Systeme
CVS_RSH           - Remote-Shell-Programm: sollte heute immer ssh sein
CVS_SERVER        - CVS-Server
CVS_SERVER_SLEEP- Server-Wartezeit nur für Debug
CVSIGNORE         - Muster für zu ignorierende Dateien
CVSREAD           - Schreibschutz für update und checkout
CVSUMASK          - Zugriffsrechte für lokale Repositories
HOME              - Homedirktory
PATH              - Pfad für ausführbare Kommandos
TMP               - Direktory für temp-Files
```



```
Administrationskommandos
------------------------

 cvs [<globale Optionen>] admin [Administrationsoptionen] files...

        -b[rev]     Einen Zweig 'rev' als Standard festlegen.
        -c string   Änderung des Kommentarstrings.
        -e[users]   Löschen von Nutzern. Nutzer stehen in einer durch
                    Kommatas getrennten Liste.
        -I          Gehe in den interaktiven Modus.
        -k kflag    Standardmodus für Schlüsselwortersetzung:
                      kv   (Default) Substitue keyword and value.
                      kvl  Substitue keyword, value, and locker
                           (if any).
                      k    Substitue keyword only.
                      o    Preserve original string.
                      b    Like o, but mark file as binary.
                      v    Substitue value only.
        -l[rev]    Die angegebenen Version  'rev' sperren
        -L         Striktes Sperren einschalten.
        -m rev:msg  Protokollnachricht der Version 'rev' ändern in 'msg'
        -n tag[:[rev]]  Den symbolischen Namen 'tag' mit den Files der
                        Version 'rev' verknüpfen. wenn 'tag' existiert
                        wird ein Fehler angezeigt.
        -N tag[:[rev]]  Wie -n aber wenn 'rev' schon existiert,  wird
                        weitergemacht.
```

```
-o range   Eine Version, die durch range beschrieben wird, wird
           endgültig gelöscht.
   rev1::rev2  Between rev1 and rev2, excluding rev1 and rev2.
   rev::       After rev on the same branch.
   ::rev       Before rev on the same branch.
   rev         Just rev.
   rev1:rev2   Between rev1 and rev2, including rev1 and rev2.
   rev:        rev and following revisions on the same branch.
   :rev        rev and previous revisions on the same branch.
-q         Keine Protokollnachrichten ausgeben (quiet-modus).
-s state[:rev]  Status der Version 'rev' in 'stat' ändern.
-t[file]   Den beschreibenden Text in CVS-Dateien auf die
           angegebenen Datei festlegen.
-t-text  Den beschreibenden Text in CVS-Dateien auf den
           angegebenen Text 'text' festlegen.
-u[rev]    Angegebenen Version entsperren, wenn nichts angegeben,
           wird die aktuelle entsperrt.
-U         Striktes Sperren ausschalten.
```



```
cvs [<globale Optionen>]  import [-b zweig] [-d] [-I muster] [-k kfalg] [-m
         [-W name] modul hersteller version

         Importieren eines kompletten Verzeichnisses als neuer Modul in das
         Repository. Neuer Kode, der bisher nicht überprüft wurde kann in
         das Repository eingegliedert werden.

           -b zweig    - Importieren eines Herstellerzweiges.
           -d          - Verwende ZEitstempel der Dateiänderung der Dateien
                         anstelle der aktuellen Systemzeit.
           -I muster   - Erkennungsmuster für zu ignorierende Dateien.
           -m infotext - Information infotext für Protokolleintrag.
           -W name     - Name des CVS-Wrappers.
           modul       - Name des neuen Moduls im Repository.
           hersteller  - Herstellerkennung.
           version     - Versionshinweis.


cvs [ -d dirctory ] init

         Initialisieren des Repository. Einzige Option -d für die Definition
         von CVS-Root. Wird die Option nicht angegeben, wird die Umgebungsva
         CVSROOT benutzt.
```

```
Files im Repsitory
------------------

checkoutlist  - verwaltet Files
commitinfo    - Konfiguriert commit-Informationen
config        - Konfiguration des Repository
cvsignore     - Erkennungsmuster für Dateien, die übergangen werden sollen
cvswrappers   - Festlegung von Standardoptionen für bestimmte CVS-Kommandos
                für bestimmte Dateien
editinfo      - RCS, Angaben für Protokolldatei-Editor
history       - Protokoll aller Tätigkeiten über dem Repository
loginfo       - Behandlung von Protokollinformationen
modules       - Verzeichnisse, die im Toplevel-Directory von CVS liegen heiße
                module. Zusätzlich können in modules logische Module
                beschrieben werden.
notify        - Enthält Informationen über die Versendung von Nachrichten für
                watch
rcsinfo       - Textschablone für commit und import
taginfo       - Verwaltungsinformationen für tag und rtag
verifymsg     - Zur Prüfung von Protokolleinträgen
readers       - Nutzer mit nur Leserechten
writers       - Nutzer mit Lese- und Schreibrechten, wenn sie nicht in reader
```



```
SVN - Subversion (Next Generation Open Source Version Control)
==============================================================

Subversion Kommandos - Übersicht
--------------------------------

   svn
           Kommandozeilen Client-Programm
   svnversion
           Programm zum Erzeugen eines Statusreports für eine Kopie
   svnlook
           Tool zur Inspektion eines Subversion-Repository
   svnadmin
           Tool zur Administration des Subversion-Repository
   svndumpfilter
           Tool zur Erzeugung eines Dumpfiles
   mod-dav_svn
           Module für Apache-Server
   mod-authz_svn
           Module für Apache-Server
   svnserve
           Standalone Server, auch mit SSH nutzbar

   Hilfe:
       man svn
       svn help
       svn help <subcommando>
```

```
Arbeiten mit svn-Hilfen
-----------------------
svn help
---------

Aufruf: svn UNTERBEFEHL [Optionen] [Parameter]
Geben Sie 'svn help UNTERBEFEHL' ein, um Hilfe zu einem Unterbefehl
zu erhalten.

Die meisten Unterbefehle akzeptieren Datei und/oder Verzeichnisparameter,
wobei die Verzeichnisse rekursiv durchlaufen werden. Wenn keine Parameter
angegeben werden, durchläuft der Befehl das aktuelle Verzeichnis rekursiv.

Verfügbare Unterbefehle:
   add
   blame (praise, annotate, ann)
   cat
   checkout (co)
   cleanup
   commit (ci)
   copy (cp)
   delete (del, remove, rm)
   diff (di)
   export
   help (?, h)
   import
   info
   list (ls)
   log
   merge
```



```
   mkdir
   move (mv, rename, ren)
   propdel (pdel, pd)
   propedit (pedit, pe)
   propget (pget, pg)
   proplist (plist, pl)
   propset (pset, ps)
   resolved
   revert
   status (stat, st)
   switch (sw)
   update (up)

Subversion ist ein Programm zur Versionskontrolle.
Für weitere Informationen siehe: http://subversion.tigris.org/
```

```
svn help update
---------------

update (up): Aktualisiert die Arbeitskopie mit Änderungen aus dem Projektarch
Aufruf: update [PFAD...]

  Ist keine Revision angegeben, wird die Arbeitskopie auf den aktuellen
  Stand der HEAD-Revision gebracht. Ansonsten wird die Arbeitskopie mit der
  durch -r angegebenen Revision synchronisiert.
  Für jedes aktualisierte Objekt wird eine Zeile mit einem Buchstaben für die
  Aktion ausgegeben. Diese haben die folgenden Bedeutungen

    A  Added    - Hinzugefügt
    D  Deleted  - Gelöscht
    U  Updated  - Aktualisiert
    C  Conflict - Konflikt
    G  merGed   - Zusammengeführt

  Ein Buchstabe in der ersten Spalte symbolisiert eine Aktualisierung der
  Datei, während Aktualisierungen der Dateieigenschaften in der zweiten
  Spalte angezeigt werden.
  Ein »B« in der dritten Spalte zeigt an, dass die Sperre für die Datei
  aufgebrochen oder gestohlen wurde.

  ..........
```



```
Gültige Optionen:

  -r [--revision] arg      : ARG (manche Befehle akzeptieren auch Wertebereic
                             Ein Revisions Parameter kann sein:
                                NUMBER       Revisionsnummer
                                "{" DATE "}" Revision zum Startdatum
                                "HEAD"       Neueste im Projektarchiv
                                "BASE"       Basisrevision der Arbeitskopie
                                "COMMITTED"  Letzte übertragene Revision bei
                                               oder vor BASE
                                "PREV"       Letzte Revision vor COMMITTED
  -N [--non-recursive]     : Nicht rekursiv hinabsteigen
  --depth PAR              : begrenzt Operation durch Tiefe PAR (»empty«,
                            »files«, »immediates« oder »infinity«)
  -q [--quiet]             : So wenig wie möglich ausgeben
  --diff3-cmd arg          : Verwende ARG als Merge Programm
  --force                  : Durchführung des Befehls erzwingen
     ....

Globale Optionen:

  --username arg           : Benutzername ARG angeben
  --password arg           : Passwort ARG angeben
  --no-auth-cache          : Anmeldeinformation nicht zwischenspeichern
  --non-interactive        : Keine interaktiven Rückfragen ausgeben
  --config-dir arg         : Benutzerkonfigurationsdateien aus dem Verzeichni
    ....
```

```
Kurzfassung Subversion-Nutzung
------------------------------


Möglichkeiten des Zugriffs auf ein Repository für Nutzer:

    file:///pfad        - lokales Filesystem

    http://host/pfad    - Zugriff über http und WebDAV
                          über Apache-Server

    https://host/pfad   - Zugriff über http und WebDAV
                          über Apache-Server aber mit SSL

    svn://host/pfad     - Zugriff über svnserve-Server

    svn+ssh://host/pfad - Zugriff über ssh und svn,
                          wie svn aber über ssh-Tunnel
```



```
Lokale Nutzung von Subversion - lokales Repository bei einem Nutzer

  Erzeugen eines leeren Repository für Subversion (lokal)

      mkdir -p /home/bell/Subversion
      chgrp svnuser /home/bell/Subversion                # *)
      chmod g+ws /home/bell/Subversion                   # *)
      svnadmin create /home/bell/Subversion
      chmod -R g+ws /home/bell/Subversion                 # *)
      ls /home/bell/Subversion
           conf/  dav/  db/  format  hooks/  locks  README.txt

  Importieren einer Anfangsversion in das SVN-Repository  (lokal)

      #
      # Quelldirectory ist in:  /home/bell/Src/Einleitung
      #
      svn import /home/bell/Src/Einleitung       \
                  file:///home/bell/Subversion/Einleitung -m "Initialzustand"

      chmod -R g+ws /home/bell/Subversion                # *)
      # Repository befindet sich in /home/bell/Subversion/Einleitung
      #

  *)  Notwendig, wenn fremde Nutzer auf das Repository
      zugreifen sollen.  UNIX-File-Zugriffsrechte beachten!!!
      Alle Nutzer müssen in der Gruppe svnuser sein!!!!
```

Arbeiten mit einem lokalen Repository

```
    # neue Sandbox erzeugen
    mkdir Sandbox

    # Füllen der Sandbox
    cd Sandbox
    svn checkout file:///home/bell/Subversion/Einleitung

    # Inhalt der Sandbox
    ls Einleitung
       Makefile  sysconf2.c  sysconf.c

    # Veränderungen vornehmen und testen
    cd Einleitung
    vi sysconf.c
    make
    ./sysconf

    # Senden der Änderung
    svn commit
    # nur sysconf.c wird übertragen, die eventuell beim Testen
    # neu erzeugten Files kommen nicht ins Repository
```


Das Gleiche entfernt mittels svn+ssh
Repository ist bei einem Nutzer als Server.
Der Zugriff erfolgt remote die Zugriffsrechte haben.

Erstellen des Repository auf dem Server!!!

```
    mkdir /home/bell/Subversion
    chgrp svnuser /home/bell/Subversion               # *)
    chmod g+ws /home/bell/Subversion                  # *)
    svnadmin create /home/bell/Subversion
    chmod -R g+ws /home/bell/Subversion               # *)
```

Übertragen einer Anfangsversion ins Repository (remote)

```
    # Quellen in /home/bell/Src/Einleitung
    cd /home/bell/Src/Einleitung
    svn import .  svn+ssh://localhost/home/bell/Subversion/Einleitung  \
                 -m "Initialzustand"
    # evnetuell lokal auf dem Server nachbessern:
    chmod -R g+ws /home/bell/Subversion               # *)
```

***) Notwendig, wenn fremde Nutzer auf das Repository zugreifen sollen. UNIX-File-Zugriffsrechte beachten!!! Alle Nutzer müssen in der Gruppe svnuser sein!!!!**

Arbeiten mit den Daten mittels svn+ssh (remote)

```
# Erzeugen eines Buddelkastens
mkdir Sandbox
# Ab in den Buddelkasten
cd Sandbox
# Spielzeug holen
svn checkout \
    svn+ssh://localhost/home/bell/Subversion/Einleitung
# Anschauen was man bekommen hat
ls Einleitung
# neues Spielzeug ausprobieren
cd Einleitung
# Spielzeug modifizieren und schauen ob es noch funktioniert
vi sysconf2.c      # ändern der Daten
make
# hoffentlich schöneres Spielzeug zurückgeben
# Senden der Änderung
svn commit
```


Das Gleiche mit Apache2 – Repository wird auf einem WWW-Server verwaltet

Vorbereiten des Apache-Servers (SuSE 11.4)

```
Module dav, dav_svn, authz_svn  aktivieren
    ( /etc/sysconfig/apache2 :
             APACHE_MODULES=" ... dav dav_svn authz_svn ...)
SVN-Location in Konfiguration einfügen
    ( /etc/apache2/conf.d/subversion.conf)
    <IfModule mod_dav_svn.c>
         <Location /Einleitung>
             DAV svn
             SVNPath /srv/svn/repos/Einleitung
             AuthType Basic
             AuthName "Authorization SVN"
             AuthUserFile /srv/svn/passwdfile
             Require valid-user
         </Location>
    </IfModule>
Direktory erzeugen:
    mkdir -p /srv/svn/repos/Einleitung
    svnadmin create --fs-type fsfs /srv/svn/repos/Einleitung
    chown -R wwwrun /srv/svn/repos/Einleitung
```

Erstinitialisierung schon als entfernter Nutzer

```
svn import /home/bell/Src/Einleitung    \
        http://localhost/Einleitung/  -m "Initialwert"
```

Anzeigen:

```
svn ls http://localhost/Einleitung
svn log http://localhost/Einleitung
```

Auschecken, bearbeiten, einchecken:

```
svn checkout http://localhost/Einleitung
cd Einleitung
vi sysconf.c
svn commit
svn log http://localhost/Einleitung
```

Im Browser unter

```
http://localhost/Einleitung/
```

anzeigbar.

GIT – Stupid Content Tracker

GIT wurde von Linus Torwalds ursprünglich für die Verwaltung der LINUX-Kernel-Quellen entworfen und sollte viel besser sein, als die bis dahin benutzten Versionsverwaltungssysteme. Inzwischen beteiligt sich eine größere Gruppe von Entwicklern – über 50 primäre Authoren – an der Weiterentwicklung von GIT.

Wesentliche Merkmale von GIT

- Verteilte Entwicklung, jeder Entwickler hat sein eigenes vollständiges Repository, dadurch auch viele Duplikate.
- Unterstützung für nicht lineare Entwicklung (Verzweigungen und Zusammenführungen möglich)
- Krypthografische Authentifikation der History des Projektes
- Code-Reviews einfacher durchführbar
- Effiziente Unterstützung für sehr große Projekte
- Toolkit Design – viele kleine Werkzeuge gehören zu Git
- Es gibt Übergangsmöglichkeiten zu anderen Versionsverwaltungssystemen

```
Dokumentationen
---------------

Im Netz:
    http://git-scm.com/                    - GIT Homepage
    http://gitref.org/                     - GIT Reference
    http://git-scm.com/course/svn.html     - GIT für SVN-Nutzer
    http://www.kernel.org/pub/software/scm/git/docs/gittutorial.html
                                           - kurzes GIT Tutorial
    http://www.kernel.org/pub/software/scm/git/docs/everyday.html
                                           - Die 20 wichtigsten Kommandos

Manuals:
    git              - GIT mit allen Unterkommandos
    git-help         - GIT Help-Funktion
    gitk             - GIT Repository-Browser
    gittutorial
    gitcore-tutorial
    gittutorial-2    - Tutorials

git selbst:
    git help xxxx    - GIT Funktion xxxx

Mit Firefox lokale Files lesen (ab SuSE 11.4):
    firefox file:///usr/share/doc/packages/git-core/
    firefox file:///usr/share/doc/packages/git-core/user-manual.html
    firefox file:///usr/share/doc/packages/git-core/everyday.html
```



```
Programmübersicht
-----------------

   git                - The stupid content tracker
                        Enhält alle Unterkommandos

   gitk               - The git repository browser

   git-cvsserver      - A CVS server emulator for git

   git-shell          - Restircted login shell for GIT-only SSH access

   git-receive-pack   - Receive what is pushed into the repository

   git-upload-archive - Send archive back to git-archive

   git-upload-pack    - Send objects packed back to git-fetch-pack
```

Das git-Kommando

```
> git


usage: git [--version] [--exec-path[=<path>]] [--html-path]
           [-p|--paginate|--no-pager] [--no-replace-objects]
           [--bare] [--git-dir=<path>] [--work-tree=<path>]
           [-c name=value] [--help]
           <command> [<args>]


Die allgemein verwendeten Git-Kommandos sind:

   add        stellt Dateiinhalte zur Eintragung bereit
   bisect     Findet über eine Binärsuche die Änderungen, die einen Fehler
              verursacht haben
   branch     Zeigt an, erstellt oder entfernt Zweige
   checkout   Checkt Zweige oder Pfade im Arbeitszweig aus
   clone      Klont ein Projektarchiv in einem neuen Verzeichnis
   commit     Trägt Änderungen in das Projektarchiv ein
   diff       Zeigt Änderungen zwischen Versionen, Version und Arbeitszweig,
              etc. an
   fetch      Lädt Objekte und Referenzen von einem anderen Projektarchiv
              herunter
   grep       Stellt Zeilen dar, die einem Muster entsprechen
   init       Erstellt ein leeres Git-Projektarchiv oder initialisiert ein
              bestehendes neu
```



```
   log        Zeigt Versionshistorie an
   merge      Führt zwei oder mehr Entwicklungszweige zusammen
   mv         Verschiebt oder benennt eine Datei, ein Verzeichnis, oder eine
              symbolische Verknüpfung um
   pull       Fordert Objekte von einem externen Projektarchiv an und führt
              sie mit einem anderen Projektarchiv oder einem lokalen Zweig
              zusammen
   push       Aktualisiert externe Referenzen mitsamt den verbundenen Objekte
   rebase     Baut lokale Versionen auf einem aktuellerem externen Zweig neu
              auf
   reset      Setzt die aktuelle Zweigspitze (HEAD) zu einem spezifizierten
              Zustand zurück
   rm         Löscht Dateien im Arbeitszweig und von der Bereitstellung
   show       Zeigt verschiedene Arten von Objekten an
   status     Zeigt den Zustand des Arbeitszweiges an
   tag        Erzeugt, listet auf, löscht oder verifiziert ein mit GPG
              signiertes Markierungsobjekt

Siehe 'git help <Kommando>' für weitere Informationen zu einem spezifischen
Kommando
```

```
Erste Schritte mit git (lokal)

Grundeinstellungen für eine Person
     git config --global user.name "Musterfrau"
     git config --global user.email "Musterfrau@mailadresse.de"
  Es wird ~/.gitconfig angelegt.

Hilfe lesen:
   git help       # Allgemeine Informationen über Hilfen
   git help log   # lesen der Hilfe für log

Einrichten des ersten GIT-Repositories

    mkdir ~/Git/Einleitung     # Mein allgeines Git-Verzeichnis
    cd    ~/Git/Einleitung
    git init --share=group       # GIT mit Gruppenzugriffsrechten
                                 # initialisieren
    #                  locales leeres Repository entsteht
    cp   ~/Src/Einleitung .   # einfüllen meiner Files
    git add .       # hinzufügen aller aktueller Files zur Verwaltung
    #               Achtung die Files sind noch nicht im Repository
    git commit  -a  # Erstmalig Übertragung der Files in Repository
    #
    #   das Repository befindet sich unter
    #         ./.git

    Hier können wir auch gleich arbeiten.
```



```
Benutzung  im Master

    cd ~/Git/Einleitung
    vi sysconf2.c
    git commit sysconf2.c
    vi sysconf2.c
    vi sysconf.c
    git commit sysconf.c sysconf2.c
    # oder
    git commit -a    # alle geänderten Files werden übertragen
```

Arbeiten mit mehreren Nutzern

```
   neuer Nutzer (tbell)
     cd    # /home/tbell/
     mkdir Sandbox
     cd Sandbox
     # hierfuer sind Zugriffsrechte auf /home/bell/Git/Einleitung notwendig
     #
     git clone /home/bell/Git/Einleitung Einleitung.neu
     cd Einleitung.neu
     vi sysconf.c
     git commit -a
     git log
     vi Bemerkung
     git add Bemerkung
     git commit Bemerkung

   Datenübertragung neuer Nutzer (tbell) -->  Master (bell)
   Master (bell):
     cd ~/Git/Einleitung
     # holen vom neuen Nutzer
     git pull /home/tbell/Sandbox/Einleitung.neu master
     vi Bemerkung
     git commit -a

   Datenübertragung Master --> neuer Nutzer
   Neuer Nutzer (tbell)
     cd ~/Sandbox/Einleitung.neu
     # Holen vom Master
     git pull /home/bell/Git/Einleitung master
```


Arbeiten mit Branches

```
     cd ~/Git/Einleitung
     # Branch "spielen" erzeugen
     git branch spielen
     # Anzeigen aller Branches
     git branch
     git checkout spielen  # Wechseln nach spielen
     git branch
     vi  Spielen
     git add Spielen    # in spielen neues File Spielen
     git commit -a      # Uebertragen
     ls
     git checkout master  # wechseln nach master
     ls                  # Spielen fehlt
     git merge spielen
     ls                  # Spielen vorhanden

  weitere Kommandos
     git log    #E History
     git show   <hash-wert>
            Änderung genau anschauen
     git show   HEAD    # letzte Änderung anzeigen
     git show   spielen # letzte Änderung des Branches spielen

     git tag version1
     git branch -D spielen # Branch spielen wegwerfen
```

GIT mit zentralem Server

1. **leeres GIT-Repository auf dem zentralem GIT-Server einrichten**

2. **Arbeiten mit zentralem GIT-Repository**

```
git config --global http.sslVerify false    #
git config --global user.name "Jan-Peter Bell"
git config --global user.email "bell@informatik.hu-berlin.de"
git config --global push.default matching
cd Git
# holen leeres Repository
git clone https://bell@git.informatik.hu-berlin.de/repos/UNIX/Socket
cd Socket
# füllen der Daten in das lokale Direktory
cp -r ../Orginal/Socket/* .
git commit -a   # aktualisieren lokales Repository
git push origin master    # uebertragen der Anfangsversion zum Server
vi asdfdsa.c      # editieren
git commit -a   # aktualisieren lokales Repository
git push origin master  # uebertragen der Änderung zum Server
```