Versionskontrolle mit SVK

Ingo Blechschmidt
<iblech@web.de>

LUGA

1. Februar 2006

Inhalt

Versionskontrolle

- Generell: Verfolgen von Versionen/Änderungen
- „Was habe ich damals geändert?“
- „Was hat `$PERSON` damals geändert?“

Grundidee

- Dateiverwaltung durchs Versionskontrollsystem
- Damit möglich:
  - Rückkehr zu früheren Ständen
  - Genaue Buchführung („wieso habe ich das gleich nochmal geändert?“)
  - Mehrgleisige Entwicklung (Zweige (branches))

Mögliche Einsatzzwecke

- Programmierung
- `/etc`
- `$WICHTIGE_ARBEIT`, `$WICHTIGES_DOKUMENT`

Geschichte

Vor 1972 **cp**
1982 RCS
1992 CVS
2002 Subversion (SVN)
2003 SVK

cp

```
$ cp datei datei.2006-01-31  # Backup
$ $EDITOR datei
$ ...
# Oh! Irgendwas ging schief!
$ mv datei.2006-01-31 datei  # Revert
```

Geschichte

Vor 1972 cp
1982 **RCS**
1992 CVS
2002 Subversion (SVN)
2003 SVK

RCS

```
$ ci   datei  # Checkin
$ co   datei  # Checkout
$ rlog datei  # Log ansehen
```

Aber: Keine Möglichkeit der Zusammenarbeit

Geschichte

Vor 1972 cp
1982 RCS
1992 **CVS**
2002 Subversion (SVN)
2003 SVK

CVS

```
$ cvs ci  datei  # Checkin
$ cvs co  datei  # Checkout
$ cvs log datei  # Log ansehen
```

- Aufsatz auf RCS (zuerst pure Shellskripte, später C)
- (Teils große) Probleme beim Umgang mit Verzeichnissen und Umbenennung
- Geringe Performance

Geschichte

Vor 1972 cp
1982 RCS
1992 **CVS**
2002 Subversion (SVN)
2003 SVK

CVS

"You hate it and then it hates back"

```
$ cat -n cvs/src/server.c
5878        goto i_hate_you;
...
5898    i_hate_you:
5899        printf ("I HATE YOU\n");
5900        fflush (stdout);
```

Geschichte

Vor 1972 cp
1982 RCS
1992 CVS
2002 **Subversion (SVN)**
2003 SVK

Subversion

```
$ svn ci  datei  # Checkin
$ svn co  datei  # Checkout
$ svn log datei  # Log ansehen
```

- Neuentwicklung
- Versionierte Verzeichnisse; Umbenennungen
- Atomare Commits
- Anzeige von Unterschieden (diffs) auch ohne Server

Geschichte

Vor 1972 cp
1982 RCS
1992 CVS
2002 **Subversion (SVN)**
2003 SVK

Subversion

"Disks are cheap, bandwidth is expensive"

Alternativen

- Perforce (schnell, kommerziell, zentraler Aufbau)
- Bitkeeper (...)
- Darcs (noch langsam)
- Arch (langsam, “insist[s] on some concrete in your brain”)
- Bazaar-NG (von Canonical (Ubuntu))
- Monotone, Aegis, ...

Entwicklung

- RCS: 1982, SVN: 2002
- Voraussetzungen für Open-Source-Entwicklung:
  - Fähige Programmierer
  - Freude oder Not (oder Geld)
- Probleme:
  - Versionskontrollprogrammierung macht wenig Spaß
  - Abfinden mit vorhandenen Lösungen

Entwicklung

- Initiator: Chia-liang Kao (clkao)
- "Screw it! I am going to take one year off and make my life easier in the future" – clkao, September 2003

Mottos von SVK

- "Disks are cheap, bandwidth is *very* expensive"
- Faulheit, Do What I Mean (DWIM)

Design

- Zugriff auf unterschiedliche Versionskontrollsysteme (Subversion, git, CVS, Perforce, ...)
- Verteilung, Dezentralität:
  Kein zentraler Server, flexible Patch-Verwaltung
- `http://perlcabal.org/~autrijus/svk-overview.png`
- Lokales Repository auf Subversion-Basis
  (Performance! Robustheit!)

Arbeitsweise beim Umgang mit entfernten Repositories

1. Mirrorn eines entfernten Repositories
2. Arbeiten im lokalen Mirror, Internetverbindung nicht erforderlich
3. Zurückschieben der Änderungen ins entfernte Repository

Design

- Zugriff auf unterschiedliche Versionskontrollsysteme (Subversion, git, CVS, Perforce, ...)
- Verteilung, Dezentralität:
  Kein zentraler Server, flexible Patch-Verwaltung
- `http://perlcabal.org/~autrijus/svk-overview.png`
- Lokales Repository auf Subversion-Basis
  (Performance! Robustheit!)

Arbeitsweise beim Umgang mit entfernten Repositories

1. Mirrorn eines entfernten Repositories
2. Arbeiten im lokalen Mirror, Internetverbindung nicht erforderlich
3. Zurückschieben der Änderungen ins entfernte Repository

Praxis: Single-User-SVK

1. Importieren eines vorhandenen Verzeichnisbaums

```
$ svk import --to-checkout \
    //grtz /home/iblech/grtz
```

2. Arbeiten
3. Ende eines Arbeitsblocks; Einchecken (checkin, commit)

```
$ svk ci
```

4. „Wieso hab ich das doch gleich gemacht?“

```
$ svk log foo.yml
```

5. „Wie war das doch nochmal vor einer Woche?“

```
$ svk update -r '{YYYY-MM-DD}'
# Oder, kürzer:
$ svk up -r '{YYYY-MM-DD}'
```

Praxis: Single-User-SVK

1. Importieren eines vorhandenen Verzeichnisbaums

```
$ svk import --to-checkout \
    //grtz /home/iblech/grtz
```

2. Arbeiten
3. Ende eines Arbeitsblocks; Einchecken (checkin, commit)

```
$ svk ci
```

4. „Wieso hab ich das doch gleich gemacht?“

```
$ svk log foo.yml
```

5. „Wie war das doch nochmal vor einer Woche?“

```
$ svk update -r '{YYYY-MM-DD}'
# Oder, kürzer:
$ svk up -r '{YYYY-MM-DD}'
```

Praxis: Single-User-SVK

1. Importieren eines vorhandenen Verzeichnisbaums

```
$ svk import --to-checkout \
    //grtz /home/iblech/grtz
```

2. Arbeiten
3. Ende eines Arbeitsblocks; Einchecken (checkin, commit)

```
$ svk ci
```

4. „Wieso hab ich das doch gleich gemacht?“

```
$ svk log foo.yml
```

5. „Wie war das doch nochmal vor einer Woche?“

```
$ svk update -r '{YYYY-MM-DD}'
# Oder, kürzer:
$ svk up -r '{YYYY-MM-DD}'
```

Praxis: Single-User-SVK

1. Importieren eines vorhandenen Verzeichnisbaums

```
$ svk import --to-checkout \
    //grtz /home/iblech/grtz
```

2. Arbeiten
3. Ende eines Arbeitsblocks; Einchecken (checkin, commit)

```
$ svk ci
```

4. „Wieso hab ich das doch gleich gemacht?“

```
$ svk log foo.yml
```

5. „Wie war das doch nochmal vor einer Woche?“

```
$ svk update -r '{YYYY-MM-DD}'
# Oder, kürzer:
$ svk up -r '{YYYY-MM-DD}'
```

Kommandoübersicht

`svk ci`	Einchecken
`svk log`	Log ansehen
`svk update -r ...`	Zu bestimmter Version zurückkehren
`svk update`, `svk up`	Zu neuester Version zurückkehren
`svk diff`	Unterschiede (diff) anzeigen
`svk add ...`	Datei oder Verzeichnis zum Repository hinzufügen
`svk rm ...`	Datei oder Verzeichnis löschen
`svk cp ... ...`	Datei oder Verzeichnis kopieren
`svk mv ... ...`	Datei oder Verzeichnis verschieben/umbenennen

Siehe auch

- `http://svk.elixus.org/`
- `http://svkbook.elixus.org/`
- `http://opensource.fotango.com/~clkao/svk-intro/start.html`
  svk – Version Control without the Headaches (clkao)
- `http://utsl.gen.nz/talks/svn-svk/slides/start.html`
  Upgrading your development to svk and svl via svn (Sam Vilain)

Siehe auch

- `http://svk.elixus.org/`
- `http://svkbook.elixus.org/`
- `http://opensource.fotango.com/~clkao/svk-intro/start.html`
  svk – Version Control without the Headaches (clkao)
- `http://utsl.gen.nz/talks/svn-svk/slides/start.html`
  Upgrading your development to svk and svl via svn (Sam Vilain)

Fragen?

Bonus-Slides

4 Anhang
- Revisionen bei Subversion und SVK
- Zweige, Tags und Trunk
- Zugriff auf entfernte Repositories mit SVK

Revisionen bei Subversion und SVK

- Markierung eines bestimmten Zeitpunkts durch eine Revisionsnummer
- Erfassung immer *aller* Dateien durch eine Revisionsnummer

Beispiel

1. Aktuelle Revision: 42; vorhandene Dateien: `grtz`, `baka`
2. Modifikation von `grtz`, `svk ci`
3. Nun aktuelle Revision: 43
4. Revision von `grtz`: 43
5. Revision von `baka`: trotz Ausbleiben von Änderungen an `baka` ebenfalls 43 (Gegensatz zu CVS!)

Revisionen bei Subversion und SVK

- Markierung eines bestimmten Zeitpunkts durch eine Revisionsnummer
- Erfassung immer *aller* Dateien durch eine Revisionsnummer

Beispiel

1. Aktuelle Revision: 42; vorhandene Dateien: `grtz`, `baka`
2. Modifikation von `grtz`, `svk ci`
3. Nun aktuelle Revision: 43
4. Revision von `grtz`: 43
5. Revision von `baka`: trotz Ausbleiben von Änderungen an `baka` ebenfalls 43 (Gegensatz zu CVS!)

Zweige, Tags und Trunk

- Bei Subversion und SVK *keine* spezielle Behandlung von Zweigen, Tags und Trunk
- Stattdessen: Zweige, Tags und Trunk nur als Verzeichnisse:

`//grtz/trunk`	Trunk
`//grtz/tags`	Verzeichnis für Tags
`//grtz/branches`	Verzeichnis für Zweige

- Keine Sonderbehandlung dieser Verzeichnisse!

Zweige, Tags und Trunk

1. Anlegen der Verzeichnisstruktur

```
$ svk mkdir //grtz/trunk
$ svk mkdir //grtz/tags
$ svk mkdir //grtz/branches
```

2. Arbeiten in *//grtz/trunk/*

3. Taggen des aktuellen Trunks als *release-3.141*

```
$ svk cp //grtz/trunk //grtz/tags/release-3.141
```

4. Weiterarbeiten in *//grtz/trunk/*

5. Erzeugen des Zweigs foobs aus...

```
# ...dem aktuellen Trunk
$ svk cp //grtz/trunk //grtz/branches/foobs
# ...release-3.141
$ svk cp \
    //grtz/tags/release-3.141 //grtz/branches/foobs
```

Zweige, Tags und Trunk

1. Anlegen der Verzeichnisstruktur

```
$ svk mkdir //grtz/trunk
$ svk mkdir //grtz/tags
$ svk mkdir //grtz/branches
```

2. Arbeiten in *//grtz/trunk/*
3. Taggen des aktuellen Trunks als *release-3.141*

```
$ svk cp //grtz/trunk //grtz/tags/release-3.141
```

4. Weiterarbeiten in *//grtz/trunk/*
5. Erzeugen des Zweigs foobs aus...

```
# ...dem aktuellen Trunk
$ svk cp //grtz/trunk //grtz/branches/foobs
# ...release-3.141
$ svk cp \
    //grtz/tags/release-3.141 //grtz/branches/foobs
```

Zweige, Tags und Trunk

1. Anlegen der Verzeichnisstruktur

```
$ svk mkdir //grtz/trunk
$ svk mkdir //grtz/tags
$ svk mkdir //grtz/branches
```

2. Arbeiten in *//grtz*/trunk/
3. Taggen des aktuellen Trunks als *release-3.141*

```
$ svk cp //grtz/trunk //grtz/tags/release-3.141
```

4. Weiterarbeiten in *//grtz*/trunk/
5. Erzeugen des Zweigs foobs aus...

```
# ...dem aktuellen Trunk
$ svk cp //grtz/trunk //grtz/branches/foobs
# ...release-3.141
$ svk cp \
    //grtz/tags/release-3.141 //grtz/branches/foobs
```

Zugriff auf entfernte Repositories mit SVK

1. Mirrorn und Auschecken des entfernten Repositories
   ```
   $ svk cp svn://svn.openfoundry.org/pugs pugs
   ```
2. Arbeiten (auch ohne Internetverbindung)
3. Ende eines Arbeitsblocks; lokales (!) Einchecken (ebenfalls ohne Internetverbindung)
   ```
   $ svk ci
   ```
4. Hochschieben der Änderungen zum entfernten Repository
   ```
   # Pro lokalem Commit ein entfernter Commit:
   $ svk push
   # Oder alle lokalen Commits in einem Commit
   # zum Server schicken:
   $ svk push -l
   ```
5. Änderungen aus dem entfernten Repository holen
   ```
   $ svk pull
   ```

Zugriff auf entfernte Repositories mit SVK

1. Mirrorn und Auschecken des entfernten Repositories
   ```
   $ svk cp svn://svn.openfoundry.org/pugs pugs
   ```
2. Arbeiten (auch ohne Internetverbindung)
3. Ende eines Arbeitsblocks; lokales (!) Einchecken (ebenfalls ohne Internetverbindung)
   ```
   $ svk ci
   ```
4. Hochschieben der Änderungen zum entfernten Repository
   ```
   # Pro lokalem Commit ein entfernter Commit:
   $ svk push
   # Oder alle lokalen Commits in einem Commit
   # zum Server schicken:
   $ svk push -l
   ```
5. Änderungen aus dem entfernten Repository holen
   ```
   $ svk pull
   ```

Zugriff auf entfernte Repositories mit SVK

1. Mirrorn und Auschecken des entfernten Repositories
   ```
   $ svk cp svn://svn.openfoundry.org/pugs pugs
   ```
2. Arbeiten (auch ohne Internetverbindung)
3. Ende eines Arbeitsblocks; lokales (!) Einchecken (ebenfalls ohne Internetverbindung)
   ```
   $ svk ci
   ```
4. Hochschieben der Änderungen zum entfernten Repository
   ```
   # Pro lokalem Commit ein entfernter Commit:
   $ svk push
   # Oder alle lokalen Commits in einem Commit
   # zum Server schicken:
   $ svk push -l
   ```
5. Änderungen aus dem entfernten Repository holen
   ```
   $ svk pull
   ```

Zugriff auf entfernte Repositories mit SVK

1. Mirrorn und Auschecken des entfernten Repositories
   ```
   $ svk cp svn://svn.openfoundry.org/pugs pugs
   ```
2. Arbeiten (auch ohne Internetverbindung)
3. Ende eines Arbeitsblocks; lokales (!) Einchecken (ebenfalls ohne Internetverbindung)
   ```
   $ svk ci
   ```
4. Hochschieben der Änderungen zum entfernten Repository
   ```
   # Pro lokalem Commit ein entfernter Commit:
   $ svk push
   # Oder alle lokalen Commits in einem Commit
   # zum Server schicken:
   $ svk push -l
   ```
5. Änderungen aus dem entfernten Repository holen
   ```
   $ svk pull
   ```