

# Advanced Subversion

## *Subversion in der Praxis*

Clifford Wolf

ROCK Linux - <http://www.rocklinux.org/>

Csync2 - <http://oss.linbit.com/csync2/>

STFL - <http://www.clifford.at/stfl/>

SPL - <http://www.clifford.at/spl/>



## Einleitung

- Inhalt
- Nicht-Inhalt

Crashkurs

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

# Einleitung



# Inhalt

## Einleitung

### ● Inhalt

### ● Nicht-Inhalt

## Crashkurs

## Branching and Merging

## Vendor Branches

## Utilities

## Projektorganisation

## Seitenblicke

## Some Subversion 1.5 Features

## Referenzen

- Subversion Crashkurs
- Branching und Merging
- Vendor Trees
- Diverse Utilities
- Projektorganisation
- Subversion 1.5 Features



# Nicht-Inhalt

## Einleitung

### ● Inhalt

### ● Nicht-Inhalt

## Crashkurs

## Branching and Merging

## Vendor Branches

## Utilities

## Projektorganisation

## Seitenblicke

## Some Subversion 1.5 Features

## Referenzen

- Repository Administration
- Server side hooks
- Kommandoreferenz
- Livedemo (ausser wenn Zeit dafür bleibt ;-)



Einleitung

**Crashkurs**

- Repositorys
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

# Crashkurs



# Repositorys

Einleitung

Crashkurs

● Repositorys

- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Repositoryverwaltung mit `svnadmin`.  
Hilfe mit: `svnadmin help` bzw. `svnadmin help kommando`
- Zugriff auf Repositorys mit URLs.
- Zugriffsmethoden auf Repositorys:
  - ◆ HTTP (WebDAV/DeltaV) mit `http://` bzw. `https://` URLs (via Apache Modul)
  - ◆ SVN-Protokoll mit `svn://` URLs (via `svnserve` Daemon)
  - ◆ SSH Tunnel mit `svn+ssh://` URLs
  - ◆ Local Filesystem mit `file://` URLs
- SVN-Repositorys sind Netzwerk-Dateisysteme mit Versionierung.



# Repository Layout

Einleitung

Crashkurs

- Repositories
- Repository Layout
- Working Cops
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Die allgemeine Empfehlung für den Verzeichnisbaum in einem Subversion Repository:
  - /trunk  
.. hier passiert die hauptsächliche Entwicklung.
  - /branches / *Branch-Name*  
.. hier liegen die aktuellen Branches.
  - /tags / *Tag-Name*  
.. hier liegen die Tags.
  - /vendor / *Vendor-Tree-Name* / *Vendor-Version*  
.. hier liegen die Vendor Trees.



# Working Copys

Einleitung

Crashkurs

- Repositorys
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

## ■ Checkout - Modify - Commit

## ■ Arbeiten mit Repositorys:

- ◆ svn command-line Tool
- ◆ Diverse grafische Clients

## ■ Hilfe mit: `svn help` bzw. `svn help kommando`

## ■ Beispiel zum Erstellen einer Working-Copy:

```
svn co http://svn.clifford.at/tools/trunk  
clifford-tools
```

## ■ Updaten einer Working-Copy: `svn up`





# Basic Operations

Einleitung

**Crashkurs**

- Repositorys
- Repository Layout
- Working Copys
- **Basic Operations**
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Dateien/Verzeichnisse hinzufügen:  
`svn add filename`
- Dateien/Verzeichnisse löschen:  
`svn rm filename`
- Dateien/Verzeichnisse kopieren:  
`svn cp old-name new-name`
- Dateien/Verzeichnisse verschieben bzw. umbenennen:  
`svn mv old-name new-name`
- Dateien verändern:  
Kein eigenes Kommando notwendig.
- Änderungen zum Server schicken:  
`svn commit`



# Inspection

Einleitung

Crashkurs

- Repositorys
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Infos zu alten Versionen: `svn log`
- Änderungen zwischen Versionen anzeigen:  
`svn diff -rALT:NEU`
- Änderungen zwischen Repository und Working Copy:  
`svn diff`
- Differenzen zwischen zwei Directorys im Repository:  
`svn diff http://svnhost/prj/trunk/dir1  
http://svnhost/prj/trunk/dir2`
- Letzte Änderung zu jeder Zeile einer Datei:  
`svn blame filename`



# Branching and Merging

Einleitung

Crashkurs

- Repositories
- Repository Layout
- Working Cops
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Branches werden mit einer Kopie erstellt. Zum Beispiel:  
`svn copy Source-URL Target-URL`
- Änderungen in einem Branch können in eine Working-Copy eines anderen Branches integriert werden:  
`svn merge -r 25:66 Source-URL .`
- Eine Working-Copy kann mit dem Kommando `svn switch` zwischen Branches umgestellt werden:  
`svn switch URL .`
- Weitere merge Features:
  - ◆ Rückwärts (also z.Bsp. `-r33:32`) mergen (aka revert)
  - ◆ Differenzen zwischen zwei Directories in ein drittes
- Für ein korrektes Merging muss über die Revisions, die gebrancht werden, extra Buch geführt werden.



# Properties

Einleitung

Crashkurs

- Repositories
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Properties sind Key-Value-Paare, die Dateien sowie Revisions zugeordnet werden.
- Beispiel einer Property auf eine Datei:  
`mime-type .. Dateityp`
- Beispiel einer Property auf eine Revision:  
`svn:author .. Autorin der Revision`
- Befehle zum Arbeiten mit Properties:  
`svn pl, svn pg, svn ps, svn pe, svn pd`



# Konflikte

Einleitung

**Crashkurs**

- Repositorys
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- **Konflikte**
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Wenn mehrere Anwenderinnen und Anwender parallel Dateien ändern, kann es zu Konflikten kommen.
- Diese Konflikte werden beim Ausführen von `svn up` erkannt.
- .. und müssen händisch aufgelöst werden.
- Nach dem Auflösen eines Konfliktes:  
`svn resolved filename`



# Binärdateien

Einleitung

**Crashkurs**

- Repositories
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- **Binärdateien**
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Bei Binärdateien kann das Auflösen von Konflikten unmöglich werden.
- Deshalb müssen Binärdateien vor dem Editieren gesperrt werden:  
`svn lock filename, svn unlock filename`
- Dateien mit der `svn:needs-lock` Property sind read-only, wenn sie nicht gerade gelocked sind.



# Mixed Checkouts

Einleitung

**Crashkurs**

- Repositories
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- **Mixed Checkouts**
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Mit `svn switch` können auch Teile einer Working-Copy auf andere Repository-Pfade umgestellt werden.
- Mit `svn up` können auch Teile einer Working-Copy auf beliebige Revisions gestellt werden.
- Eine Anwendung für Mixed Checkouts ist das Bisecting von Fehlern.



# Peg-Revisions

Einleitung

Crashkurs

- Repositories
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- WC-zu-Repository Kopien

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Inhalt der README Datei in Revision 25:  

```
svn cat -r 23 http://svnhost/prj/trunk/README
```
- Inhalt jener Datei, die bei Revision 25 den Namen README hatte, in Revision 25:  

```
svn cat http://svnhost/prj/trunk/README@25
```
- Implizite Default Peg-Revisions:
  - ◆ Für Working-Copys: `BASE`
  - ◆ Für direkten Repository-Zugriff: `HEAD`





# WC-zu-Repository Kopien

Einleitung

**Crashkurs**

- Repositories
- Repository Layout
- Working Copys
- Basic Operations
- Inspection
- Branching and Merging
- Properties
- Konflikte
- Binärdateien
- Mixed Checkouts
- Peg-Revisions
- **WC-zu-Repository Kopien**

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Commit mit gleichzeitigem Branch/Tag:

```
svn cp .
```

```
http://svnhost/prj/branches/newbranch
```

- Speziell für Tags sehr wichtig.



# Branching and Merging

Einleitung

Crashkurs

**Branching and Merging**

- `svnbranch`
- `svnbranch show`
- `svnbranch branch`
- `svnbranch tag`
- `svnbranch sync`
- `svnbranch sync URL`
- `svnbranch propagate`
- `svnbranch merge URL`
- `svnbranch diff`
- `svnbranch list/graph/show`

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen



# svnbranch

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

● **svnbranch**

- svnbranch show
- svnbranch branch
- svnbranch tag
- svnbranch sync
- svnbranch sync URL
- svnbranch propagate
- svnbranch merge URL
- svnbranch diff
- svnbranch list/graph/show

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- svnbranch ist ein Shellscrip zum Verwalten von Branches.  
`http://svn.clifford.at/tools/trunk/svnbranch.sh`
- Hilfetext: `svnbranch help`
- In einem Branch werden in der Subversion Property `svnbranch:parent` das Directory und die Revision, von der gebrancht wurde, gespeichert.
- svnbranch unterstützt beim Branchen, Taggen, Mergen, Synchronisieren und Analysieren von Branches und Tags.



# svnbranch show

Einleitung

Crashkurs

Branching and Merging

● svnbranch

● **svnbranch show**

● svnbranch branch

● svnbranch tag

● svnbranch sync

● svnbranch sync URL

● svnbranch propagate

● svnbranch merge URL

● svnbranch diff

● svnbranch list/graph/show

Vendor Branches

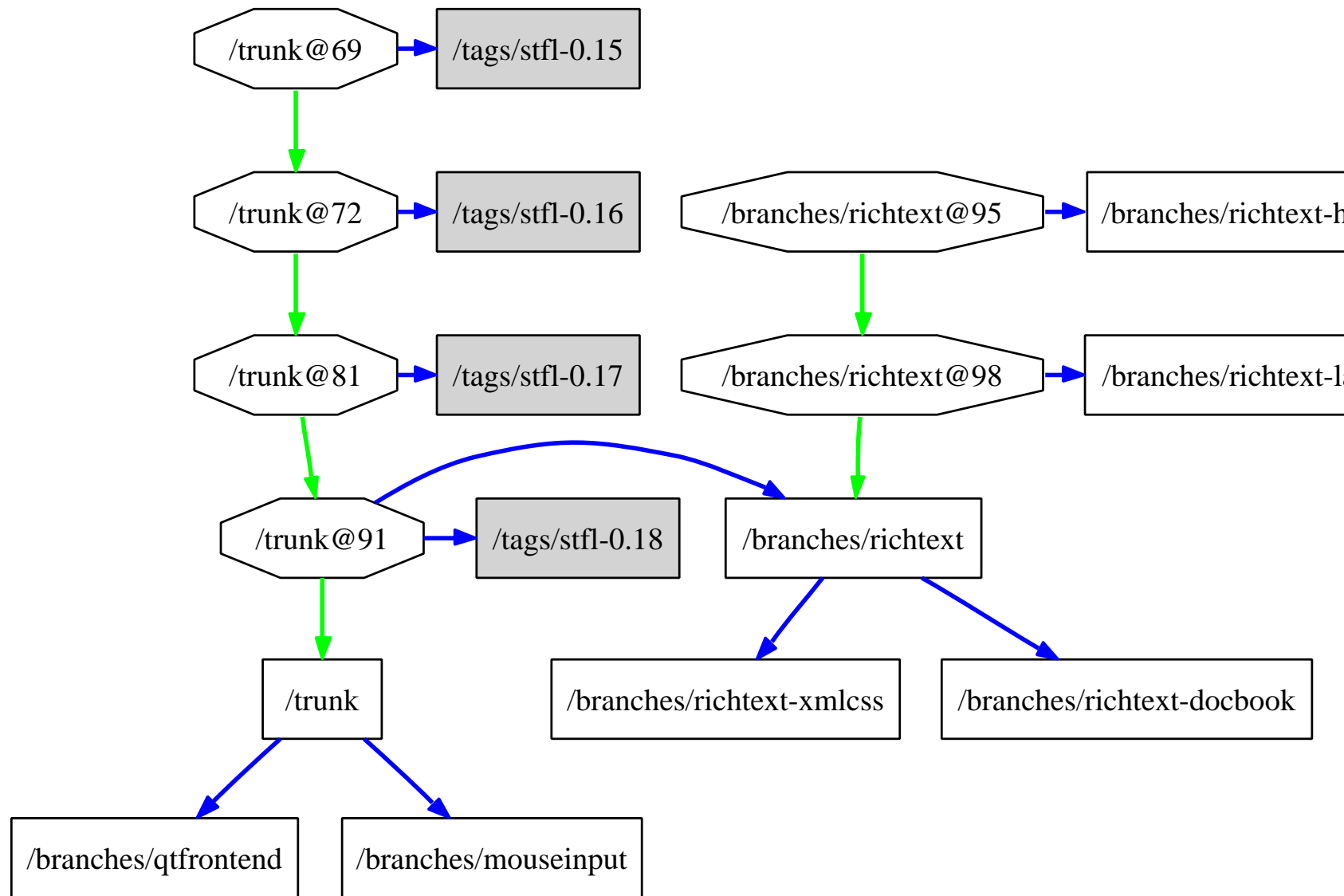
Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen





# svnbranch branch

Einleitung

Crashkurs

Branching and Merging

- svnbranch
- svnbranch show
- **svnbranch branch**
- svnbranch tag
- svnbranch sync
- svnbranch sync URL
- svnbranch propagate
- svnbranch merge URL
- svnbranch diff
- svnbranch list/graph/show

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Erzeugt einen neuen Branch.
- Wird in einer Working-Copy des Directorys gestartet, das gebrancht werden soll:

`svnbranch branch Target-URL`

- Erzeugt eine Kopie (`svn cp`)
- Setzt die Property `svnbranch:parent` (`svn ps`)
- Löscht die Property `svnbranch:tag` (`svn pd`)



# svnbranch tag

Einleitung

Crashkurs

Branching and Merging

- svnbranch
- svnbranch show
- svnbranch branch
- **svnbranch tag**
- svnbranch sync
- svnbranch sync URL
- svnbranch propagate
- svnbranch merge URL
- svnbranch diff
- svnbranch list/graph/show

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Wird in einer Working-Copy des Directorys gestartet, das getaggt werden soll.

`svnbranch tag Target-URL`

- Erzeugt eine Kopie (`svn cp`)
- Setzt die Property `svnbranch:parent` (`svn ps`)
- Setzt die Property `svnbranch:tag` (`svn pd`)
- Bei Tags werden Änderungen im Parent nicht synchronisiert.



# svnbranch sync

Einleitung

Crashkurs

Branching and Merging

- svnbranch
- svnbranch show
- svnbranch branch
- svnbranch tag
- **svnbranch sync**
- svnbranch sync URL
- svnbranch propagate
- svnbranch merge URL
- svnbranch diff
- svnbranch list/graph/show

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Wird in einer Working-Copy eines Branches gestartet.

- Übernimmt Änderungen im Parent seit dem Branch / letzten Syncen in den Branch.

```
svnbranch sync  
svnbranch commit
```

- Mergt die Änderungen vom Parent (`svn merge`)
- Aktualisiert die Property `svnbranch:parent` (`svn ps`)



# svnbranch sync URL

Einleitung

Crashkurs

Branching and Merging

- svnbranch
- svnbranch show
- svnbranch branch
- svnbranch tag
- svnbranch sync
- **svnbranch sync URL**
- svnbranch propagate
- svnbranch merge URL
- svnbranch diff
- svnbranch list/graph/show

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Wird in einer Working-Copy eines Branches gestartet.
- Stellt einen Branch auf einen Parent um.
- Die Differenzen zwischen altem und neuem Parent werden gemergt.

```
svnbranch sync URL  
svnbranch commit
```

- Mergt die Änderungen von altem zu neuem Parent (`svn merge`)
- Aktualisiert die Property `svnbranch:parent` (`svn ps`)





# svnbranch propagate

Einleitung

Crashkurs

Branching and Merging

- svnbranch
- svnbranch show
- svnbranch branch
- svnbranch tag
- svnbranch sync
- svnbranch sync URL
- **svnbranch propagate**
- svnbranch merge URL
- svnbranch diff
- svnbranch list/graph/show

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Wird in der Working-Copy eines Parents gestartet.
- Erzeugt ein Shell-Script `propagate.sh`.
- Dieses Script synchronisiert rekursiv alle Branches des Parents.

```
svnbranch propagate  
vi propagate.sh  
sh propagate.sh
```

- Bei Konflikten wird automatisch angehalten, damit die Anwenderin den Konflikt auflösen und anschliessend committen kann.
- Ein neuerlicher Start von `propagate.sh` setzt den Prozess an jenem Punkt fort, an dem angehalten wurde.



# svnbranch merge URL

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

- [svnbranch](#)
- [svnbranch show](#)
- [svnbranch branch](#)
- [svnbranch tag](#)
- [svnbranch sync](#)
- [svnbranch sync URL](#)
- [svnbranch propagate](#)
- [svnbranch merge URL](#)
- [svnbranch diff](#)
- [svnbranch list/graph/show](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- Wird in der Working-Copy eines Parents gestartet.
- Mergt Änderungen im angegebenen Branch in den Parent.

```
svnbranch merge Branch-URL  
svn commit
```



# svnbranch diff

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

- [svnbranch](#)
- [svnbranch show](#)
- [svnbranch branch](#)
- [svnbranch tag](#)
- [svnbranch sync](#)
- [svnbranch sync URL](#)
- [svnbranch propagate](#)
- [svnbranch merge URL](#)
- [svnbranch diff](#)
- [svnbranch list/graph/show](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- Wird in einer Working-Copy eines Branches gestartet.

- Zeigt die Änderungen zum Parent an.

```
svnbranch diff
```

- Die angezeigten Änderungen beziehen sich auf jene Version des Parents die gebrancht bzw. zuletzt gesynced wurde.



# svnbranch list/graph/show

Einleitung

Crashkurs

Branching and Merging

- svnbranch
- svnbranch show
- svnbranch branch
- svnbranch tag
- svnbranch sync
- svnbranch sync URL
- svnbranch propagate
- svnbranch merge URL
- svnbranch diff
- **svnbranch list/graph/show**

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Zeigt die Struktur der Branches an.

`svnbranch show Basis-URL`

- Um alle Branches finden zu können benötigt dieses Kommando die Property `svnbranch:index` im Root-Directory des Repositorys. Zum Beispiel:

```
# svn pg svnbranch:index http://svnhost/prj  
tags/*  
branches/*
```

- Betriebsmodi:
  - ◆ `svnbranch list`: ASCII Ausgabe
  - ◆ `svnbranch graph`: Erstellt `.dot` und `.ps` Dateien
  - ◆ `svnbranch show`: Zeigt Graphen mit `kghostview` an



Einleitung

Crashkurs

Branching and Merging

**Vendor Branches**

- Worum es geht
- Wie es gemacht wird
- svnemboss
- Beispiel: Neuer Vendor Branch
- Beispiel: Vendor Branch updaten
- Patch Stacks (1/2)
- Patch Stacks (2/2)
- svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

# Vendor Branches



# Worum es geht

Einleitung

Crashkurs

Branching and Merging

**Vendor Branches**

● **Worum es geht**

● Wie es gemacht wird

● svnemboss

● Beispiel: Neuer Vendor

Branch

● Beispiel: Vendor Branch  
updaten

● Patch Stacks (1/2)

● Patch Stacks (2/2)

● svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Man möchte externe Sourcen (z.Bsp. eine Library) ins eigene Projekte integrieren.
- Man möchte in dieser Library Änderungen vornehmen.
- Gleichzeitig möchte man den neuen Versionen der Library folgen.



# Wie es gemacht wird

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

● Worum es geht

● Wie es gemacht wird

● svnemboss

● Beispiel: Neuer Vendor

Branch

● Beispiel: Vendor Branch

updaten

● Patch Stacks (1/2)

● Patch Stacks (2/2)

● svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Dazu importiert man die Library in ein eigenes Verzeichnis.  
Zum Beispiel: `/vendor/foolib/foolib-1.0`
- Dieses Verzeichnis brancht man in das eigene Projekt hinein.  
Zum Beispiel: `/trunk/foolib`
- Neue Versionen der Library bekommen jeweils ein eigenes Verzeichnis.  
Zum Beispiel: `/vendor/foolib/foolib-1.1`
- Die Änderungen zwischen diesen Verzeichnissen können in den eigenen Branch hineingemergt werden.



# svnemboss

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

- Worum es geht
- Wie es gemacht wird
- **svnemboss**
- Beispiel: Neuer Vendor Branch
- Beispiel: Vendor Branch updaten
- Patch Stacks (1/2)
- Patch Stacks (2/2)
- svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Neue Versionen des Vendor Trees müssen mit der alten "verwandt" sein.
- Sonst besteht die Differenz zwischen den beiden Trees aus deletes der alten und adds der neuen Dateien. Und das lässt sich nicht mit lokalen Änderungen mergen.
- Das Tool svnemboss ist hilfreich, um Änderungen im upstream Vendor Tree im lokalen Subversion abzubilden.

`http://svn.clifford.at/tools/trunk/svnemboss.sh`

- Mit svnemboss ist es möglich, in einer Subversion Working-Copy alle Änderungen vorzunehmen, sodass sie einem angegebenen Directory entspricht:

```
svnemboss Master-Path WC-Path
```

```
svn commit WC-Path
```





# Beispiel: Neuer Vendor Branch

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

- Worum es geht
- Wie es gemacht wird
- svnemboss
- Beispiel: Neuer Vendor Branch
- Beispiel: Vendor Branch updaten
- Patch Stacks (1/2)
- Patch Stacks (2/2)
- svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

```
$ svn import ~/foolib-1.0-orig-src \  
http://svnhost/prj/vendor/foolib/foolib-1.0
```

```
$ svn co http://svnhost/prj/vendor/foolib/foolib-1.0  
foolib-wc
```

```
$ cd foolib-wc
```

```
$ svnbranch branch http://svnhost/prj/trunk/foolib
```

```
$ svn switch http://svnhost/prj/trunk/foolib
```

```
<local changes>
```

```
$ svn commit
```



# Beispiel: Vendor Branch updaten

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

- Worum es geht
- Wie es gemacht wird
- svnemboss
- Beispiel: Neuer Vendor Branch
- Beispiel: Vendor Branch updaten

- Patch Stacks (1/2)
- Patch Stacks (2/2)
- svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

```
$ svn co http://svnhost/prj/vendor/foolib/foolib-1.0
    foolib-wc
```

```
$ cd foolib-wc
```

```
$ svnbranch branch \
    http://svnhost/prj/vendor/foolib/foolib-1.1
```

```
$ svn switch \
    http://svnhost/prj/vendor/foolib/foolib-1.1
```

```
$ svnemboss ~/foolib-1.1-orig-src .
```

```
$ svn commit
```

```
$ svn switch http://svnhost/prj/trunk/foolib
```

```
$ svnbranch sync \
    http://svnhost/prj/vendor/foolib/foolib-1.1
```

```
$ svn commit
```



# Patch Stacks (1/2)

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

- Worum es geht
- Wie es gemacht wird
- svnemboss
- Beispiel: Neuer Vendor Branch
- Beispiel: Vendor Branch updaten

● Patch Stacks (1/2)

- Patch Stacks (2/2)
- svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Manchmal hat man zusätzlich zum Vendor Tree noch weitere 3rd-Party Patches.
- Diese möchte man in der Regel voneinander, von den original Sourcen und von den eigenen Änderungen getrennt halten.
- Das geht einfach, indem man Branches von Branches macht. Jeder Patch hat dann seinen eigenen Branch, wobei die eigenen Änderungen im letzten dieser Branches liegen.
- Da `svnbranch sync` das Wechseln der Parent-URL erlaubt, können Patches beliebig hinzugefügt, entfernt oder ausgetauscht werden.



# Patch Stacks (2/2)

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

- Worum es geht
- Wie es gemacht wird
- svneboss
- Beispiel: Neuer Vendor Branch
- Beispiel: Vendor Branch updaten
- Patch Stacks (1/2)
- Patch Stacks (2/2)
- svn:externals

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

## ■ Beispielstack:

```
/vendor/linux-kernel/linux-2.6.24.4  
/vendor/linux-kernel-patches/denx-eldk-4.1  
/vendor/linux-kernel-patches/xilinuxfb  
/vendor/linux-kernel-patches/unionfs-2.3.2  
/vendor/linux-kernel-patches/dmatransfer  
/trunk/linux-kernel-source
```

- Man kann auch die Subversion Merge-Features benutzen, um Patches für unterschiedliche Versionen des Vendor Trees auf eine gemeinsame Version zu bringen.
- Ebenso kann man die Subversion Merge-Features benutzen, um Konflikte zwischen den Patches aufzulösen.



# svn:externals

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

- Worum es geht
- Wie es gemacht wird
- svnemboss
- Beispiel: Neuer Vendor Branch
- Beispiel: Vendor Branch updaten
- Patch Stacks (1/2)
- Patch Stacks (2/2)
- **svn:externals**

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Externe Sourcen, die nicht lokal verändert werden sollen, können mittels `svn:externals` Property eingebunden werden.
- Die `svn:externals` Property wird auf das übergeordnete Directory gesetzt. Also z.Bsp. auf `/trunk` für `/trunk/foolib`.
- Solche Externals werden automatisch von der Originalquelle ausgecheckt (`svn co`) und geupdated (`svn up`).
- Commits gehen nicht automatisch in die external-Verzeichnisse.
- Beispiel:  

```
$ svn pg svn:externals http://svnhost/prj/trunk  
foolib http://host1/foolib/tags/foolib-1.1  
barlib -r531 http://host2/barlib/trunk
```



[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

**[Utilities](#)**

- [svncache \(1/2\)](#)
- [svncache \(2/2\)](#)
- [svnfind](#)
- [svngrep](#)
- [svnlist](#)
- [svnignore](#)
- [svnpurge](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

# Utilities



# svncache (1/2)

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

● svncache (1/2)

● svncache (2/2)

● svnfind

● svngrep

● svnlist

● svnignore

● svnpurge

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Bei grossen Projekten (z.Bsp. linux kernel, gcc, firefox) können Operationen im Repository recht langsam werden.
- Klassisches Beispiel: Diff zwischen zwei Branches.
- Mit `svncache.sh` können read-only Operationen über einen lokalen Mirror durchgeführt werden.
- `svncache` benutzt als Backend das `svnsync` Tool aus dem Subversion Paket.

<http://svn.clifford.at/tools/trunk/svncache.sh>



## svncache (2/2)

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

**Utilities**

● [svncache \(1/2\)](#)

● **[svncache \(2/2\)](#)**

● [svnfind](#)

● [svngrep](#)

● [svnlist](#)

● [svnignore](#)

● [svnpurge](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

```
$ svnadmin create file:///data/svncache/clifford-tools
```

```
$ ln -s /bin/true \  
    /data/svncache/clifford-tools/hooks/pre-revprop-
```

```
$ svnsync init file:///data/svncache/clifford-tools  
    http://svn.clifford.at/tools
```

```
$ echo http://svn.clifford.at/tools \  
    file:///data/svncache/clifford-tools \  
    >> /etc/svnchache
```

```
$ svncache sync
```

```
<cd to a http://svn.clifford.at/tools wc>
```

```
$ svncache log -v -r300:600
```





# svnfind

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

● [svncache \(1/2\)](#)

● [svncache \(2/2\)](#)

● [svnfind](#)

● [svngrep](#)

● [svnlist](#)

● [svnignore](#)

● [svnpurge](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- `svnfind` ist ein Wrapper um `find`, der alle `.svn` Directorys ignoriert.

- Beispiel:

```
svnfind . -type f -exec file '{}' \;
```

<http://svn.clifford.at/tools/trunk/svnfind.sh>



# svngrep

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

● svncache (1/2)

● svncache (2/2)

● svnfind

● **svngrep**

● svnlist

● svnignore

● svnpurge

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- **svngrep** ist ein Wrapper um **grep**, der alle **.svn** Directories ignoriert.

- **Beispiel:**  
`svngrep -ri copyright .`

`http://svn.clifford.at/tools/trunk/svngrep.sh`

- **svngrep** benötigt **grep** in der Version 2.5.2 oder höher, oder alternativ folgenden Patch applied:

`http://svn.clifford.at/tools/trunk/patches/grep-2.5.1a-excl-dir.patch`



# svnlist

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

● [svncache \(1/2\)](#)

● [svncache \(2/2\)](#)

● [svnfind](#)

● [svngrep](#)

● [svnlist](#)

● [svnignore](#)

● [svnpurge](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- `svnlist` erzeugt eine Liste aller unter der Kontrolle von Subversion stehenden Dateien im Arbeitsverzeichnis.
- In einer frischen Working-Copy entspricht das der Ausgabe von `svnfind -type f`.
- Anwendungsbeispiel: Automatisches Generieren von Makefile Dependencies.

<http://svn.clifford.at/tools/trunk/svnlist.sh>



# svnignore

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

● svncache (1/2)

● svncache (2/2)

● svnfind

● svngrep

● svnlist

● **svnignore**

● svnpurge

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Hilft beim Verwalten der `svn:ignore` Property's.

- Anwendungsbeispiel zum Ignorieren einzelner Dateien:

```
svnignore add foolib/foolib.so  
svnignore del foolib/foolib.so
```

- Anwendungsbeispiel zum Ignorieren ganzer Muster:

```
svnignore radd 'linux-kernel-source/*.ko'  
svnignore rdel 'linux-kernel-source/*.ko'
```

<http://svn.clifford.at/tools/trunk/svnignore.sh>



# svnpurge

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

● svncache (1/2)

● svncache (2/2)

● svnfind

● svngrep

● svnlist

● svnignore

● **svnpurge**

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

- Macht alle lokalen Änderungen in der Working-Copy rückgängig und löscht alle Dateien, die nicht unter der Kontrolle von Subversion stehen.
- **ACHTUNG:** Mit diesem Tool kann man sich auch leicht ins eigene Knie schießen!
- Anwendungsbeispiel: Besonders effektiver Ersatz für `make distclean`.

`http://svn.clifford.at/tools/trunk/svnpurge.sh`



# Projektorganisation

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

- Warum Subversion?
- Subversion Manager
- Weitere Tipps

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)



# Warum Subversion?

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

● **Warum Subversion?**

● Subversion Manager

● Weitere Tipps

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- Es entspricht dem bekannten Konzept eines hierarchischen Dateisystems.
- Es gibt gute grafische Clients und gute IDE-Integration.
- Durch den zentralen Ansatz ist es leicht in Backup-Systeme zu integrieren.
- Es gibt ausreichend einsteigerfreundliche Literatur.



# Subversion Manager

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

● [Warum Subversion?](#)

● [Subversion Manager](#)

● [Weitere Tipps](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- Es hat sich bewährt eine Verantwortliche für Subversion zu bestimmen.
- Diese Person sollte ausreichend Zeit haben um sich mit den Subversion-spezifischen Problemen im Projekt eingehend zu beschäftigen.
- Ausserdem sollte diese Person regelmässige Commits einfordern und Changelogs, etc. reviewen.  
(Das Problem Nummer 1 in der Praxis ist, dass Leute einfach nicht comitten!)
- Komplexere Aufgaben (Vendor Branches, etc.) können an den Subversion Manager delegiert werden.





# Weitere Tipps

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

● [Warum Subversion?](#)

● [Subversion Manager](#)

● [Weitere Tipps](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- Wer sich bei einer Operation unsicher ist, kann mit `svnsync` eine Replik des Repositorys anlegen um zu testen.
- Manchmal ist es sinnvoll, einfach Checkout auf einem Fileserver liegen zu haben und dort regelmässig per Cron-Job automatisch zu comitten.
- .. speziell in Umgebungen mit Nicht-Programmiererinnen ist das schmerzfreier als den Umgang mit Subversion zu vermitteln.



[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

- Svnmerge.py
- svn\_load\_dirs.pl
- svm
- svk
- SubMaster

[Some Subversion 1.5 Features](#)

[Referenzen](#)

# Seitenblicke



# Svnmerge.py

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

● [Svnmerge.py](#)

● [svn\\_load\\_dirs.pl](#)

● [svm](#)

● [svk](#)

● [SubMaster](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- `Svnmerge.py` ist ein Konkurrent zu meinem `svnbranch` Tool.
- `Svnmerge.py` konzentriert sich auf das reine Merge-Tracking. Es ist z.Bsp. nicht möglich den Branch-Graphen zu verändern. Damit sind z.Bsp. Patch-Stacks nur sehr schwer zu realisieren.
- Dafür unterstützt `Svnmerge.py` z.Bsp. von selbst das Blocken einzelner Revisions für Merges.

<http://www.orcaware.com/svn/wiki/Svnmerge.py>



# svn\_load\_dirs.pl

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

**Seitenblicke**

● [Svnmerge.py](#)

● [svn\\_load\\_dirs.pl](#)

● [svm](#)

● [svk](#)

● [SubMaster](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- `svn_load_dirs.pl` ist ein Helper Script das mit Subversion mitgeliefert wird.
- Das Script erleichtert das Verwalten von Vendor Trees, kümmert sich aber nicht um die Synchronisation mit den lokalen Branches.
- Es operiert direkt mit dem Repository und committet daher automatisch alles was es tut.
- `svn_load_dirs.pl` ist voll kompatibel mit `svnbranch`, ersetzt aber `svnemboss` für den Einsatz bei Vendor Trees.



# svm

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

● Svnmerge.py

● svn\_load\_dirs.pl

● svm

● svk

● SubMaster

Some Subversion 1.5 Features

Referenzen

- `svm` erlaubt es ein Directory von einem Repository vollständig (also inclusive History!) in ein Directory in einem anderen Repository zu replizieren.
- Auf diese Weise lassen sich z.Bsp. Vendor Trees mit der vollständigen Geschichte aus dem original Repository erzeugen.
- `svm` erlaubt ebenfalls ein Mergeback von Änderungen im neuen Repository in das ursprüngliche Repository.
- `svm` ist Teil des `SVN::Mirror` Perl Moduls.



# svk

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

**Seitenblicke**

- Svnmerge.py
- svn\_load\_dirs.pl
- svm

● **svk**

● SubMaster

[Some Subversion 1.5 Features](#)

[Referenzen](#)

■ svk ist ein Tool zur dezentralen Versionsverwaltung.

■ svk verwendet Subversion und svm als Basis.

<http://svk.elixus.org/>



# SubMaster

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

● [Svnmerge.py](#)

● [svn\\_load\\_dirs.pl](#)

● [svm](#)

● [svk](#)

● [SubMaster](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

- SubMaster ist ein Tool zur *verteilten Entwicklung*, das im ROCK-Linux-Projekt entstanden ist.
- Wie bei einem dezentralen Versionsverwaltungstool kann jede Entwicklerin unabhängig von externen Ressourcen arbeiten.
- Die einzelnen Entwicklerinnen können mit einem eigenen Kommando lokale Änderungen zu einem speziellen Server schicken.
- Dort können andere Entwicklerinnen die Änderungen über eine Weboberfläche testen und kommentieren.
- So können keine Patches in Vergessenheit geraten. Jeder Patch muss applied oder dezidiert abgelehnt werden.

<http://www.rocklinux.org/wiki/SubMaster>



Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

**Some Subversion 1.5 Features**

- Merge tracking
- Sparse Checkouts
- svn:externals improved

Referenzen

# Some Subversion 1.5 Features





# Merge tracking

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

● Merge tracking

● Sparse Checkouts

● svn:externals improved

Referenzen

- Subversion 1.5 hat nativ-merge-tracking-Funktionen, wie von `svnmerge.py` zur Verfügung gestellt.
- Merge-tracking-Daten aus `svnmerge.py` können in das Subversion-1.5-eigene Format konvertiert werden.
- Beispiel:

```
$ cd BRANCH_WORKING_COPY
$ svn merge URL_TO_TRUNK

$ cd TRUNK_WORKING_COPY
$ svn merge --reintegrate URL_TO_BRANCH
```
- Mit der Option `--record-only` können Revisions vom Mergen ausgeschlossen werden.
- `svn log` und `svn blame` wurden um die Option `-g` (`--use-merge-history`) erweitert.



# Sparse Checkouts

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

- Merge tracking
- **Sparse Checkouts**
- svn:externals improved

Referenzen

- Subversion 1.5 hat *offiziellen* Support für Sparse Checkouts.
- .. das sind Working Copys die nicht alle Dateien beinhalten.
- Anwendungsbeispiele:
  - ◆ Grosse Projekte, die man nicht vollständig auschecken möchte
  - ◆ Umsortieren von Branches und/oder Tags
- Die Kommandos `checkout`, `switch` und `up` wurden um die Option `--depth` erweitert.
- Erlaubte Werte für `--depth=...` sind: `empty`, `files`, `immediates` und `infinity`.



# svn:externals improved

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

● Merge tracking

● Sparse Checkouts

● **svn:externals improved**

Referenzen

- `svn:externals` URLs beinhalten nun Support für Peg-Revisions.
- Man kann nun in `svn:externals` auch relative Pfade angeben.
- Bei den relativen Pfaden gibt es einige besondere Features:
  - ◆ Relativ zum Repository Root:  
`^/project2/trunk`
  - ◆ In ein paralleles Repository:  
`^/../trunk`
  - ◆ Relativ zum URL Schema:  
`//example.com/svn/repos-1/project2/trunk`
  - ◆ Relativ zum Server Root:  
`/repos-1/project2/trunk`



[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

[Referenzen](#)

● [URLs \(1/2\)](#)

● [URLs \(2/2\)](#)

# Referenzen



# URLs (1/2)

Einleitung

Crashkurs

Branching and Merging

Vendor Branches

Utilities

Projektorganisation

Seitenblicke

Some Subversion 1.5 Features

Referenzen

● URLs (1/2)

● URLs (2/2)

## ■ Subversion Homepage

<http://subversion.tigris.org/>

## ■ Subversion Handbook

<http://svnbook.red-bean.com/>

## ■ Subversion 1.5 Features

[http://subversion.tigris.org/  
svn\\_1.5\\_releasenotes.html](http://subversion.tigris.org/svn_1.5_releasenotes.html)

## ■ Svnmerge.py

<http://www.orcaware.com/svn/wiki/Svnmerge.py>

## ■ SVK

<http://svk.elixus.org/>

## ■ SubMaster

<http://www.rocklinux.org/wiki/SubMaster>



# URLs (2/2)

[Einleitung](#)

[Crashkurs](#)

[Branching and Merging](#)

[Vendor Branches](#)

[Utilities](#)

[Projektorganisation](#)

[Seitenblicke](#)

[Some Subversion 1.5 Features](#)

**Referenzen**

● [URLs \(1/2\)](#)

● [URLs \(2/2\)](#)

## ■ Cliffords Tools

<http://svn.clifford.at/tools/trunk/>

## ■ Clifford Wolf

<http://www.clifford.at/>

## ■ Weitere Präsentationen

<http://www.clifford.at/papers/>

## ■ Diese Präsentation

<http://www.clifford.at/papers/2008/advsvn/>