

Distributed Software Development using Subversion and SubMaster

Infrastructure for the Bazaar

Clifford Wolf

LINBIT

<http://www.linbit.com>

Introduction

- Development Models
- Requirements
- What is Subversion (1)
- What is Subversion (2)
- Subversion is for the Cathedral
- What is SubMaster
- Components
- Mortal Kombat

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The snap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

Introduction

Introduction

● Development Models

- Requirements
- What is Subversion (1)
- What is Subversion (2)
- Subversion is for the Cathedral
- What is SubMaster
- Components
- Mortal Kombat

Using Subversion

The SubMaster Client

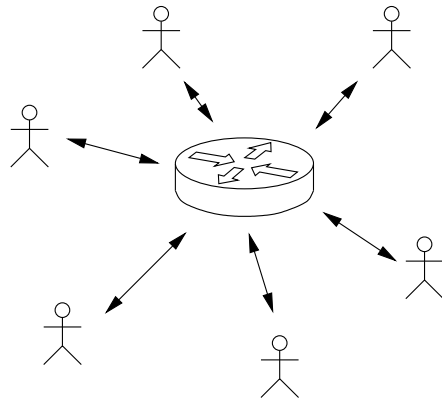
The SubMaster Server

The snap helper script

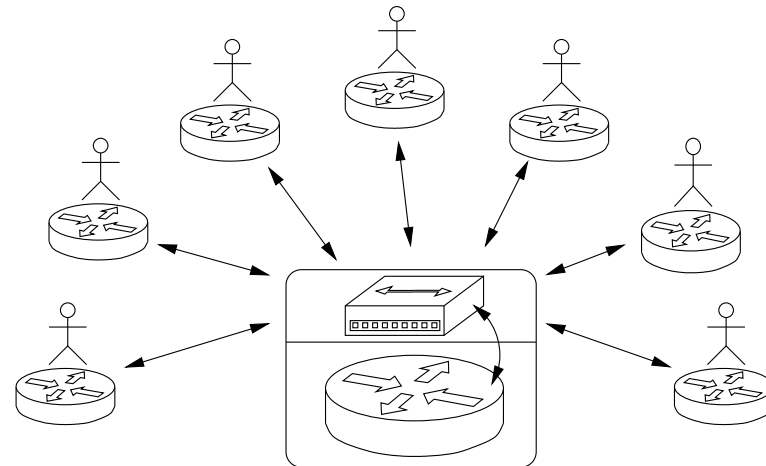
SubMaster Action Scripts

URLs and References

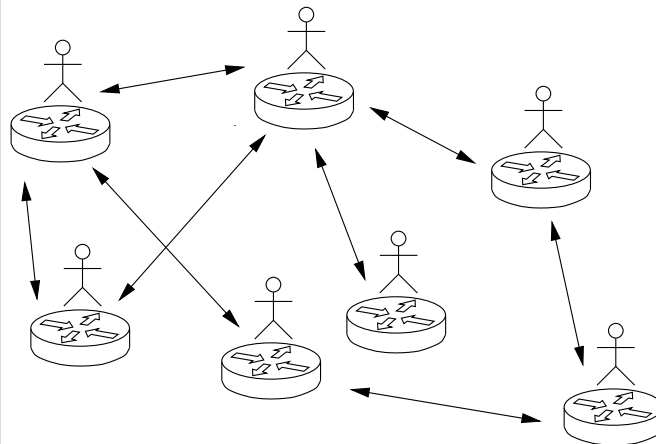
Centralized Model



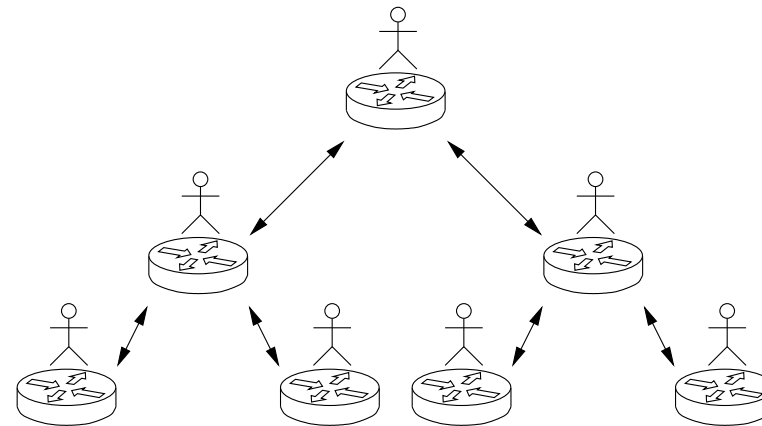
Distributed Model



Decentralized Model



Hierarchical Model



Introduction

- Development Models

● Requirements

- What is Subversion (1)

- What is Subversion (2)

- Subversion is for the Cathedral

- What is SubMaster

- Components

- Mortal Kombat

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Everyone must have the possibility to send patches
- There must be no way for patches to be lost or ignored
- Creating and sending patches must be as easy as possible
- Keeping local changes and main tree in sync must be easy
- There must be no restrictions due to licences or binary only executables
- Using the system without a GUI should be possible

What is Subversion (1)

Introduction

- Development Models
- Requirements
- What is Subversion (1)
- What is Subversion (2)
- Subversion is for the Cathedral
- What is SubMaster
- Components
- Mortal Kombat

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Subversion is a a version control system
- The entire history of a project including all branches is stored in a Database called Subversion Repository
- A Subversion Repository can also be seen as a filesystem with special capabilities
- People can check out files or directories from a Subversion Repository, change them and commit the changes back to the repository
- Some changes can be done directly on the repository without creating a working copy

What is Subversion (2)

Introduction

- Development Models
- Requirements
- What is Subversion (1)
- **What is Subversion (2)**
- Subversion is for the Cathedral
- What is SubMaster
- Components
- Mortal Kombat

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Subversion Repositories can be accessed thru the filesystem, using HTTP/WebDAV or by a special SVN:// protocol
- The tool for creating and maintaining the database behind a Subversion Repository is called 'svnadmin'
- The tool for working with the Subversion Repository and Working copies is called 'svn'
- The calling convention for 'svn' is simmilar to the calling convention for 'cvs'

Subversion is for the Cathedral

Introduction

- Development Models
- Requirements
- What is Subversion (1)
- What is Subversion (2)
- Subversion is for the Cathedral
- What is SubMaster
- Components
- Mortal Kombat

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Subversion on it's own is not good for bazaar-style development
- There is only one central repository
- Only a limited number of people has write access to the tree
- Everyone else has to send patches per email
- It's hard to keep a local tree with patches in sync with the official tree

What is SubMaster

Introduction

- Development Models
- Requirements
- What is Subversion (1)
- What is Subversion (2)
- Subversion is for the Cathedral
- **What is SubMaster**
- Components
- Mortal Kombat

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- SubMaster is a set of scripts based on Subversion targeting bazaar-style development
- Everyone has his own local Subversion repository
- SubMaster keeps the local repository in sync with the master repository preserving local changes
- SubMaster provides an infrastructure for sending patches, collecting feedback and applying patches to the master tree
- Working with SubMaster almost feels like having write access to the official tree

Introduction

- Development Models
- Requirements
- What is Subversion (1)
- What is Subversion (2)
- Subversion is for the Cathedral
- What is SubMaster
- Components
- Mortal Kombat

Using Subversion

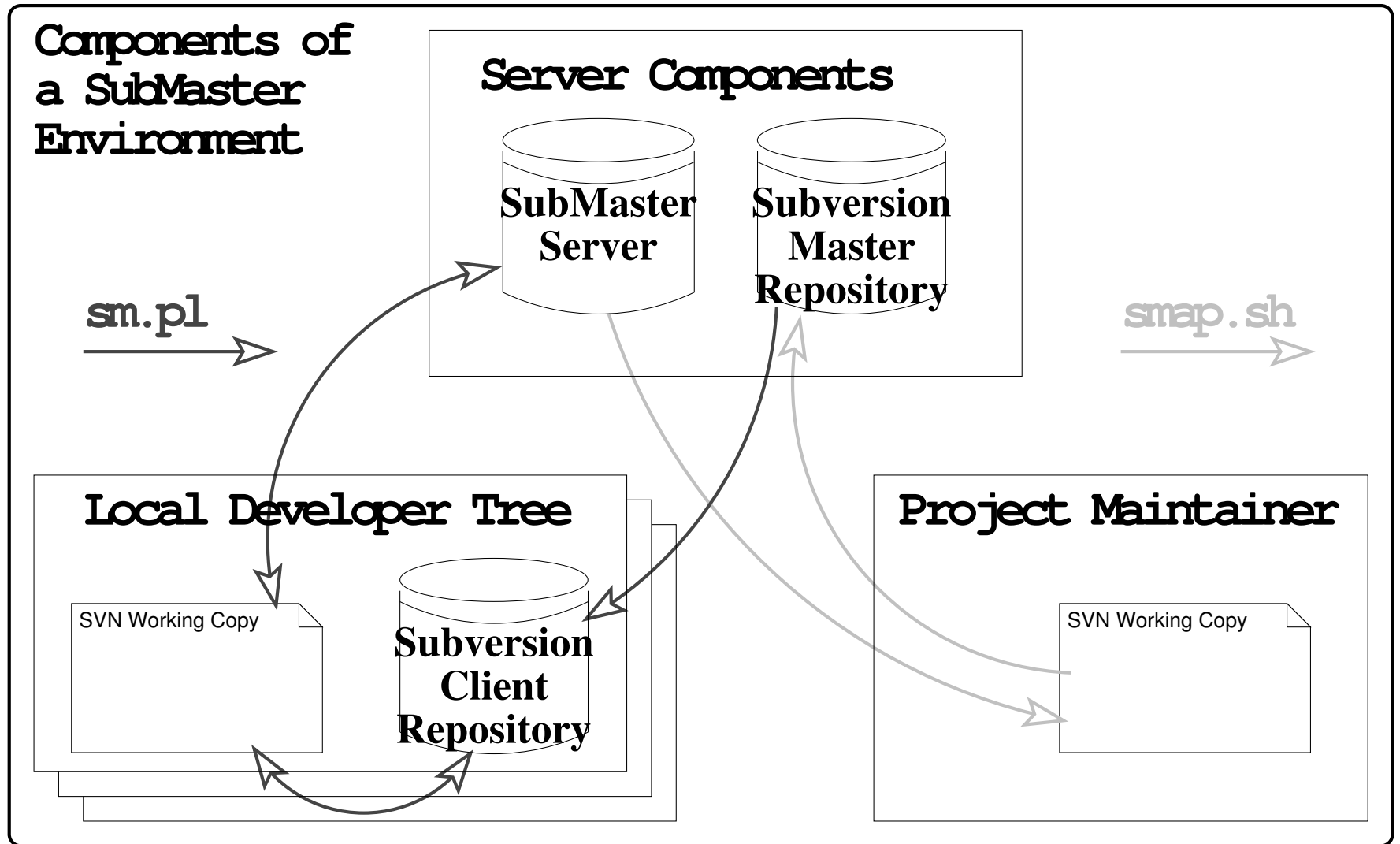
The SubMaster Client

The SubMaster Server

The smap helper script

SubMaster Action Scripts

URLs and References



Introduction

- Development Models
- Requirements
- What is Subversion (1)
- What is Subversion (2)
- Subversion is for the Cathedral
- What is SubMaster
- Components
- **Mortal Kombat**

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- In his keynote about Linux Kernel development at the german Linux Kongress, Russty Russel did talk about "Code By Confrontation".

- He called that

The Mortal Kombat Model of Software Development

- So the SubMaster tools are nothing else than

The Weapons and Battlefield of the Mortal Kombat of Software Development

[Introduction](#)

[Using Subversion](#)

- [Creating a repository](#)
- [Creating a working copy](#)
- [Making changes \(1\)](#)
- [Making changes \(2\)](#)
- [Committing changes](#)
- [Branches and tags](#)
- [Getting Help \(1\)](#)
- [Getting Help \(2\)](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The snap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

Using Subversion

Creating a repository

Introduction

Using Subversion

● Creating a repository

● Creating a working copy

● Making changes (1)

● Making changes (2)

● Committing changes

● Branches and tags

● Getting Help (1)

● Getting Help (2)

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Creating the repository itself is easy:
`svnadmin create /home/svn/repos`
- If the repository should be accessible thru HTTP/WebDAV (`http://`), you need to configure your apache to load the Subversion module and set it up.
- If the repository should be accessible thru the Subversion protocol (`svn://`), you need to set up the Subversion server
`svnserve` .

Creating a working copy

Introduction

Using Subversion

- Creating a repository
- Creating a working copy
- Making changes (1)
- Making changes (2)
- Committing changes
- Branches and tags
- Getting Help (1)
- Getting Help (2)

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Using the tool `svn` is very similar to using `cv`s .

- A Working copy can be created with:

```
svn co file:///home/svn/repos localdir  
cd localdir
```

- Note that no equivalent to `$CVSROOT` exists.
- Within the working copy, every directory has a `.svn` subdirectory containing the Subversion metadata.

Making changes (1)

Introduction

Using Subversion

- Creating a repository
- Creating a working copy
- Making changes (1)
- Making changes (2)
- Committing changes
- Branches and tags
- Getting Help (1)
- Getting Help (2)

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Modifying files:
 - ◆ Just edit them as usual with your favorite editor
 - ◆ Note that Subversion is using binary deltas and has good support for non-ascii files
- Adding files:
 - ◆ First create the new file as usual
 - ◆ Then execute `svn add filename`
- Removing files:
 - ◆ Just execute `svn rm filename`
 - ◆ The file will automatically removed by `svn`

Making changes (2)

Introduction

Using Subversion

- Creating a repository
- Creating a working copy
- Making changes (1)
- Making changes (2)
- Committing changes
- Branches and tags
- Getting Help (1)
- Getting Help (2)

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Moving/renaming files:
 - ◆ Using `svn mv` instead of `mv`
- Copying files:
 - ◆ Using `svn cp` instead of `cp`
- Making changes without a working copy:
 - ◆ Most operations can also be performed directly on the repository:
 - ◆

```
svn copy -m "Commit message" \
file:///home/svn/repos/demo2.txt \
file:///home/svn/repos/demo3.txt
```

Committing changes

[Introduction](#)

[Using Subversion](#)

- [Creating a repository](#)
- [Creating a working copy](#)
- [Making changes \(1\)](#)
- [Making changes \(2\)](#)
- [Committing changes](#)
- [Branches and tags](#)
- [Getting Help \(1\)](#)
- [Getting Help \(2\)](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The snap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

- Committing changes made in the working copy:

```
svn commit
```

- Listing status (modified, etc) of files in working copy:

```
svn status
```

- Bringing changes from the repository into the working copy:

```
svn up
```


Branches and tags

Introduction

Using Subversion

- Creating a repository
- Creating a working copy
- Making changes (1)
- Making changes (2)
- Committing changes
- **Branches and tags**
- Getting Help (1)
- Getting Help (2)

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Subversion has no explicit support for branching.
- But it has support for copying directories in O(1).
- Creating a branch is just the same as creating a copy:

```
svn copy -m "Creating branch dummy" \
    file:///home/svn/repos/trunk \
    file:///home/svn/repos/branches/dummy
```

- Merging changes from somewhere else to working copy:

```
svn merge -r70:86 \
    file:///svn/repos/branch/testing .

svn merge file:///svn/repos/branch/stable \
    file:///svn/repos/branch/testing .
```

- A “tagged version” is also just a copy. It just never gets modified.

Getting Help (1)

Introduction

Using Subversion

- Creating a repository
- Creating a working copy
- Making changes (1)
- Making changes (2)
- Committing changes
- Branches and tags
- Getting Help (1)
- Getting Help (2)

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- The command `svn help` lists all subcommands for `svn`:
`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`)

- The command `svn help subcommand` gives a detailed description

Getting Help (2)

Introduction

Using Subversion

- Creating a repository
- Creating a working copy
- Making changes (1)
- Making changes (2)
- Committing changes
- Branches and tags
- Getting Help (1)
- Getting Help (2)

The SubMaster Client

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- The command `svnadmin help` does it for `svnadmin` :
`create, deltify, dump, help (? , h), hotcopy,`
`list-dblogs, list-unused-dblogs, load,`
`lstxns, recover, rmtxns, setlog, verify`
- The command `svnadmin help subcommand` gives a detailed description
- The Subversion Handbook is a great Subversion tutorial:
<http://svnbook.red-bean.com/>

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

- Overview
- Creating a local tree
- Creating patches (1)
- Creating patches (2)
- Syncing with the main tree
- WIP Archive
- Filesystem Layout (1)
- Filesystem Layout (2)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

The SubMaster Client

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)[● Overview](#)

- Creating a local tree
- Creating patches (1)
- Creating patches (2)
- Syncing with the main tree
- WIP Archive
- Filesystem Layout (1)
- Filesystem Layout (2)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

- The SubMaster client `sm` is a simple perl script which automates the creation and management of local Subversion repositories
- The SubMaster client also automates the creation of patches and sending them upstream to the SubMaster server
- A detailed help message explaining all subcommands to `sm` is printed out by `sm help`

Creating a local tree

Introduction

Using Subversion

The SubMaster Client

● Overview

● **Creating a local tree**

● Creating patches (1)

● Creating patches (2)

● Syncing with the main tree

● WIP Archive

● Filesystem Layout (1)

● Filesystem Layout (2)

The SubMaster Server

The smap helper script

SubMaster Action Scripts

URLs and References

- Create a local copy of the master repository:

```
sm create svn://example.org/demo/trunk demo
```

- This creates two directories:

- ◆ demo The local working copy
- ◆ demo.sm Local repository and other sm data

- Create an account at the SubMaster Server:

```
w3m https://example.org/demo/submaster
```

- Add your SubMaster Login information to your SM tree:

```
vi demo.sm/SM/server.txt  
https://example.org/demo/submaster  
<username>  
<password>
```

Creating patches (1)

Introduction

Using Subversion

The SubMaster Client

- Overview
- Creating a local tree
- **Creating patches (1)**
- Creating patches (2)
- Syncing with the main tree
- WIP Archive
- Filesystem Layout (1)
- Filesystem Layout (2)

The SubMaster Server

The smap helper script

SubMaster Action Scripts

URLs and References

- Make your changes in `demo/` as usual.
- Use `svn add`, `svn rm` and `svn mv` for adding, removing and moving files or directories
- Use `sm commit` instead of `svn commit` to commit your changes to the local repository
- Sometimes an `svn up` is required before `sm commit` can succeed.

Creating patches (2)

Introduction

Using Subversion

The SubMaster Client

- Overview
- Creating a local tree
- Creating patches (1)
- **Creating patches (2)**
- Syncing with the main tree
- WIP Archive
- Filesystem Layout (1)
- Filesystem Layout (2)

The SubMaster Server

The snap helper script

SubMaster Action Scripts

URLs and References

- Use `sm queue` to list the pending changes:

```
r44 | root | 2004-01-10 12:10:58 +0100 (Sat,
Added package qprofile.
```

```
r51 | root | 2004-01-10 13:34:35 +0100 (Sat,
Qprofile: don't run deprood.
```

- Use `sm patch` to create a patch from your changes:

```
sm patch r44 r51
```

- Use `sm send` to send the patch upstream to the SubMaster Server.

Syncing with the main tree

Introduction

Using Subversion

The SubMaster Client

● Overview

● Creating a local tree

● Creating patches (1)

● Creating patches (2)

● Syncing with the main tree

● WIP Archive

● Filesystem Layout (1)

● Filesystem Layout (2)

The SubMaster Server

The smap helper script

SubMaster Action Scripts

URLs and References

- Merging changes from master tree to local tree:
 - ◆ `sm sync`
 - ◆ manually resolve conflicts, if any
 - ◆ patches being applied in master do not result in conflicts
 - ◆ `svn commit -m "SM Sync 2434:2442"`
- Full-syncing local tree to master:
 - ◆ `sm fsync`
 - ◆ This will discard all local changes
- Showing differences between local tree and master:
 - ◆ `sm diff`

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

- Overview
- Creating a local tree
- Creating patches (1)
- Creating patches (2)
- Syncing with the main tree

[● WIP Archive](#)

- Filesystem Layout (1)
- Filesystem Layout (2)

[The SubMaster Server](#)

[The snap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

- The WIP (work-in-progress) archive can be used to suspend the work on a changeset and resume it later.
- Revert in working copy and save as patch in WIP archive:
`sm wip push name r180 r181 r182`
- Apply patch in working tree and remove from WIP archive:
`sm wip pull name`
- Move patch from WIP archive to current working directory:
`sm wip co name`
- Move patch from current working directory to WIP archive:
`sm wip ci name`
- List patches in WIP archive: `sm wip`

Filesystem Layout (1)

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

- Overview
- Creating a local tree
- Creating patches (1)
- Creating patches (2)
- Syncing with the main tree
- WIP Archive

[● Filesystem Layout \(1\)](#)

● Filesystem Layout (2)

[The SubMaster Server](#)

[The snap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

- `demo/` The local Subversion working copy
- `demo.sm/* .patch` Patches waiting to be send to server
- `demo.sm/SENT/` Patches already sent to server
- `demo.sm/WIP/` Archive of work-in-progres patches
- `demo.sm/MASTER/` Exported up-to-date master tree
- `demo.sm/SVN/` Local Subversion Repository (DB)

Filesystem Layout (2)

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

- Overview
- Creating a local tree
- Creating patches (1)
- Creating patches (2)
- Syncing with the main tree
- WIP Archive
- Filesystem Layout (1)
- **Filesystem Layout (2)**

[The SubMaster Server](#)

[The snap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

- `demo/SM/master.txt` URL of main SVN repository
- `demo/SM/mrev.txt` Revision currently in MASTER/
- `demo/SM/queue.txt` Revisions in the change queue
- `demo/SM/server.txt` URL, etc. for SM Server
- `demo/SM/sync.txt` Last master rev. synced to local

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

- [Overview](#)
- [Patch Properties](#)
- [Filesystem Layout \(1\)](#)
- [Filesystem Layout \(2\)](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

The SubMaster Server

Introduction

Using Subversion

The SubMaster Client

The SubMaster Server

● Overview

● Patch Properties

● Filesystem Layout (1)

● Filesystem Layout (2)

The smap helper script

SubMaster Action Scripts

URLs and References

- Basically a cgi-based relay for patches
- Everyone can register as user
- Everyone can upload patches
- Everyone can vote for/agains and comment patches
- The tree maintainer can pull patches and apply them to the master Subversion Repository
- The tools `sm` and `smap` can communicate directly with the cgi script, no webbrowser is needed
- E-mail notifications, etc. can be implemented using action scripts

[Introduction](#)[Using Subversion](#)[The SubMaster Client](#)[The SubMaster Server](#)[● Overview](#)[● Patch Properties](#)[● Filesystem Layout \(1\)](#)[● Filesystem Layout \(2\)](#)[The smap helper script](#)[SubMaster Action Scripts](#)[URLs and References](#)

- A Unique Patch ID (such as "2004011020364615018")
- The login-name of the user who created the patch
- Current patch status (open, applied, rejected and discarded)
- The patch file itself
- The patch description (header of the patch file)
- Votes pro and contra the patch
- Other users (single-line) comments on the patch
- Auto-created additional info from the action scripts

Filesystem Layout (1)

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

● Overview

● Patch Properties

● **Filesystem Layout (1)**

● Filesystem Layout (2)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

- `password/<username>.txt`
encrypted password
- `password/<username>.super`
if this file exists, the matching user is a superuser
- `user/<username>.email`
the e-mail address of that user
- `open/<year> _<month> _<id>.open`
empty file which marks a patch as open

Filesystem Layout (2)

Introduction

Using Subversion

The SubMaster Client

The SubMaster Server

● Overview

● Patch Properties

● Filesystem Layout (1)

● Filesystem Layout (2)

The smap helper script

SubMaster Action Scripts

URLs and References

- `data/<year>/<month>/<id>.patch`
the patch file as originally sent by the user
- `data/<year>/<month>/<id>.msg`
messages attached to the patch
- `data/<year>/<month>/<id>.votes`
votes by different users
- `data/<year>/<month>/<id>.owner`
this patches owner
- `data/<year>/<month>/<id>.info`
auto-created additional info
- `data/<year>/<month>/<id>.done`
either "Applied" or "Rejected" does not exist for open patches

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The snap helper script](#)

- Overview
- Options
- Configuration
- Fixdiff

[SubMaster Action Scripts](#)

[URLs and References](#)

The snap helper script

Introduction

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

● Overview

● Options

● Configuration

● Fixdiff

SubMaster Action Scripts

URLs and References

- Helps applying patches from the SM Server
 - ◆ Download patch from server
 - ◆ Check if it applies cleanly
 - ◆ Apply the patch to the current directory
 - ◆ Run `svn add` and `svn rm` commands
 - ◆ Run `svn commit`
 - ◆ Mark patch as applied

- E.g. for maintaining the official tree
- E.g. for testing or extending patches

- Sometimes called by `sm`

- Usage: `snap [options] <patch-id>`

Introduction

Using Subversion

The SubMaster Client

The SubMaster Server

The smap helper script

● Overview

● Options

● Configuration

● Fixdiff

SubMaster Action Scripts

URLs and References

- `-pN -l -R` pass this option to the patch program
- `-d` just make a dry-run

- `-S` don't commit to svn (e.g. if this isn't a svn tree)
- `-E` edit commit message before running svn commit
- `-D` don't remove temp files after success

- `-a file` add this smap command to file after success
- `-A` don't apply anything, just do `-a`

- `-M` mark patch as applied on SM Server
- `-C` do nothing if already marked as applied

[Introduction](#)[Using Subversion](#)[The SubMaster Client](#)[The SubMaster Server](#)[The smap helper script](#)[● Overview](#)[● Options](#)[● Configuration](#)[● Fixdiff](#)[SubMaster Action Scripts](#)[URLs and References](#)

- If this is a SubMaster tree, read config from the SM config.
- Otherwise read server URL, Username and Password from `./smap.cfg`.
- It's also possible modify the parameter list in `./smap.cfg`:

```
set -- -M -I "$@"  
URL="https://www.rocklinux.net/submaster"  
USER="admin-username"  
PASS="admin-password"
```

Introduction

Using Subversion

The SubMaster Client

The SubMaster Server

The snap helper script

● Overview

● Options

● Configuration

● **Fixdiff**

SubMaster Action Scripts

URLs and References

- Can be used to create patches for patches (e.g. so they apply cleanly)

- `snap 2004040509423327913` (this fails)

- `fixdiff 2004040509423327913.patch` `co`
`package/base/linux24/lx` `_config.sh`

- Merge `package/base/linux24/lx` `_config.sh.rej`

- `fixdiff 2004040509423327913.patch` `ci`
`package/base/linux24/lx` `_config.sh`

- `→ fixdiff _2004040509423327913.patch`

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

● [Overview](#)

● [Calling Convention](#)

[URLs and References](#)

SubMaster Action Scripts

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

● Overview

● Calling Convention

[URLs and References](#)

- Action scripts are called whenever the status of a patch changes
- This can be used to send notify mails
- .. to check who should review the patch
- .. to eventually run any regression test on the patch
- The action scripts can be written in any language
- The action script for the ROCK SM is written in shell

Calling Convention

Introduction

Using Subversion

The SubMaster Client

The SubMaster Server

The smap helper script

SubMaster Action Scripts

● Overview

● Calling Convention

URLs and References

- `./action.sh` `id user new`
- `./action.sh` `id user msg 'Message Text'`

- `./action.sh` `id user vote contra`
- `./action.sh` `id user vote pro`
- `./action.sh` `id user vote delete`

- `./action.sh` `id user status rejected`
- `./action.sh` `id user status applied`
- `./action.sh` `id user status opened`
- `./action.sh` `id user status discarded`

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

- [Subversion and SubMaster](#)
- [Related Projects](#)
- [Credits](#)

URLs and References

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

● **Subversion and SubMaster**

● Related Projects

● Credits

- The Subversion Handbook:
<http://svnbook.red-bean.com/>
- The Subversion Homepage:
<http://subversion.tigris.org/>
- The SubMaster Homepage:
<http://www.rocklinux.org/submaster.html>

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

● [Subversion and SubMaster](#)

● [Related Projects](#)

● [Credits](#)

- **SVK: A decentralized system based on Subversion**
- **Transvn: A patch-scripts clone based on Subversion**
<http://alexm.here.ru/transvn/>
- **SVM: Mirror Remote Subversion Repository to local**
<http://search.cpan.org/~clkao/SVN-Mirror/>

[Introduction](#)

[Using Subversion](#)

[The SubMaster Client](#)

[The SubMaster Server](#)

[The smap helper script](#)

[SubMaster Action Scripts](#)

[URLs and References](#)

● [Subversion and SubMaster](#)

● [Related Projects](#)

● [Credits](#)

- LINBIT Information Technologies GmbH:
<http://www.linbit.com/>
- The ROCK Linux Project:
<http://www.rocklinux.org/>
- Clifford Wolf:
<http://www.clifford.at/>

<http://www.rocklinux.org/submaster.html>