Distributed Software Development using Subversion and SubMaster

Infrastructure for the Bazaar

Clifford Wolf

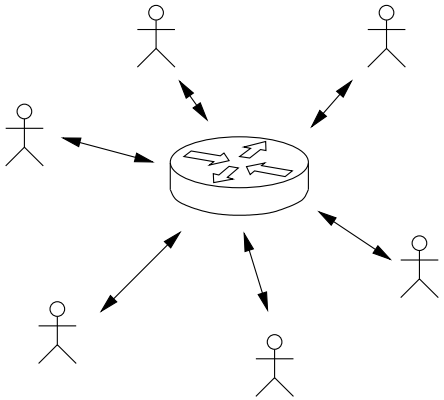
LINBIT

http://www.linbit.com

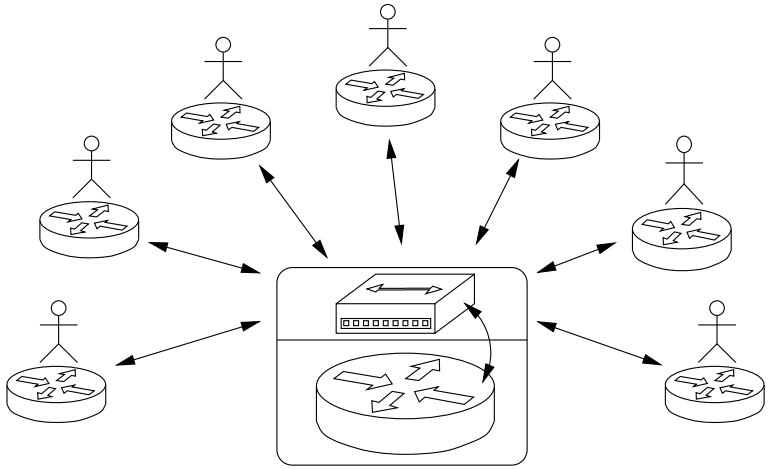
Introduction

Development Models

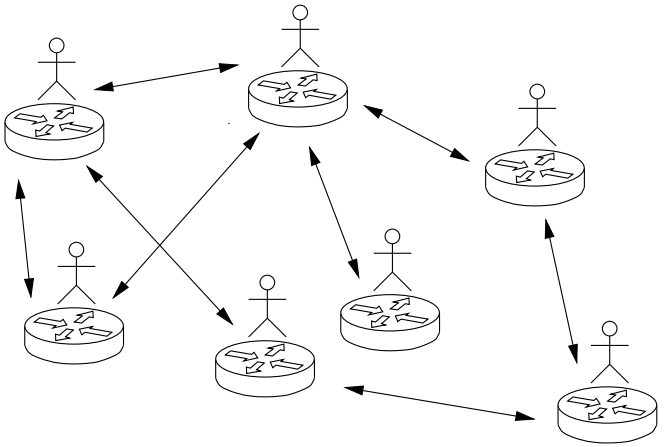
Centralized Model



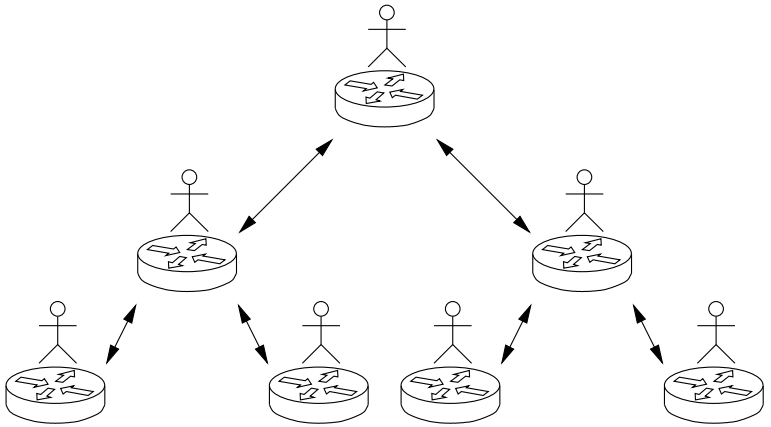
Distributed Model



Decentralized Model



Hierarchic Model



LINBIT Information Technologies GmbH

Requirements

- 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 restictions due to licences or binary only executables
- Using the system without a GUI should be possible

LINBIT Information Technologies GmbH

Requirements

- 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 restictions due to licences or binary only executables
- Using the system without a GUI should be possible

LINBIT Information Technologies GmbH

Requirements

- 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 restictions due to licences or binary only executables
- Using the system without a GUI should be possible

LINBIT Information Technologies GmbH

Requirements

- 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 restictions due to licences or binary only executables
- Using the system without a GUI should be possible

LINBIT Information Technologies GmbH

Requirements

- 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 restictions due to licences or binary only executables
- Using the system without a GUI should be possible

LINBIT Information Technologies GmbH

Requirements

- 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 restictions due to licences or binary only executables
- Using the system without a GUI should be possible

LINBIT Information Technologies GmbH

What is Subversion (1)

- 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
- Pleople 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

LINBIT Information Technologies GmbH

What is Subversion (1)

- 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
- Pleople 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

LINBIT Information Technologies GmbH

What is Subversion (1)

- 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
- Pleople 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

LINBIT Information Technologies GmbH

What is Subversion (1)

- 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
- Pleople 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

LINBIT Information Technologies GmbH

What is Subversion (1)

- 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
- Pleople 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)

- 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’

LINBIT Information Technologies GmbH

What is Subversion (2)

- 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’

What is Subversion (2)

- 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’

LINBIT Information Technologies GmbH

What is Subversion (2)

- 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’

LINBIT Information Technologies GmbH

Subversion is for the Cathedral

- 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

LINBIT Information Technologies GmbH

Subversion is for the Cathedral

- 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

LINBIT Information Technologies GmbH

Subversion is for the Cathedral

- 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

LINBIT Information Technologies GmbH

Subversion is for the Cathedral

- 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

LINBIT Information Technologies GmbH

Subversion is for the Cathedral

- 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

LINBIT Information Technologies GmbH

What is SubMaster

- 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

LINBIT Information Technologies GmbH

What is SubMaster

- 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

LINBIT Information Technologies GmbH

What is SubMaster

- 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

LINBIT Information Technologies GmbH

What is SubMaster

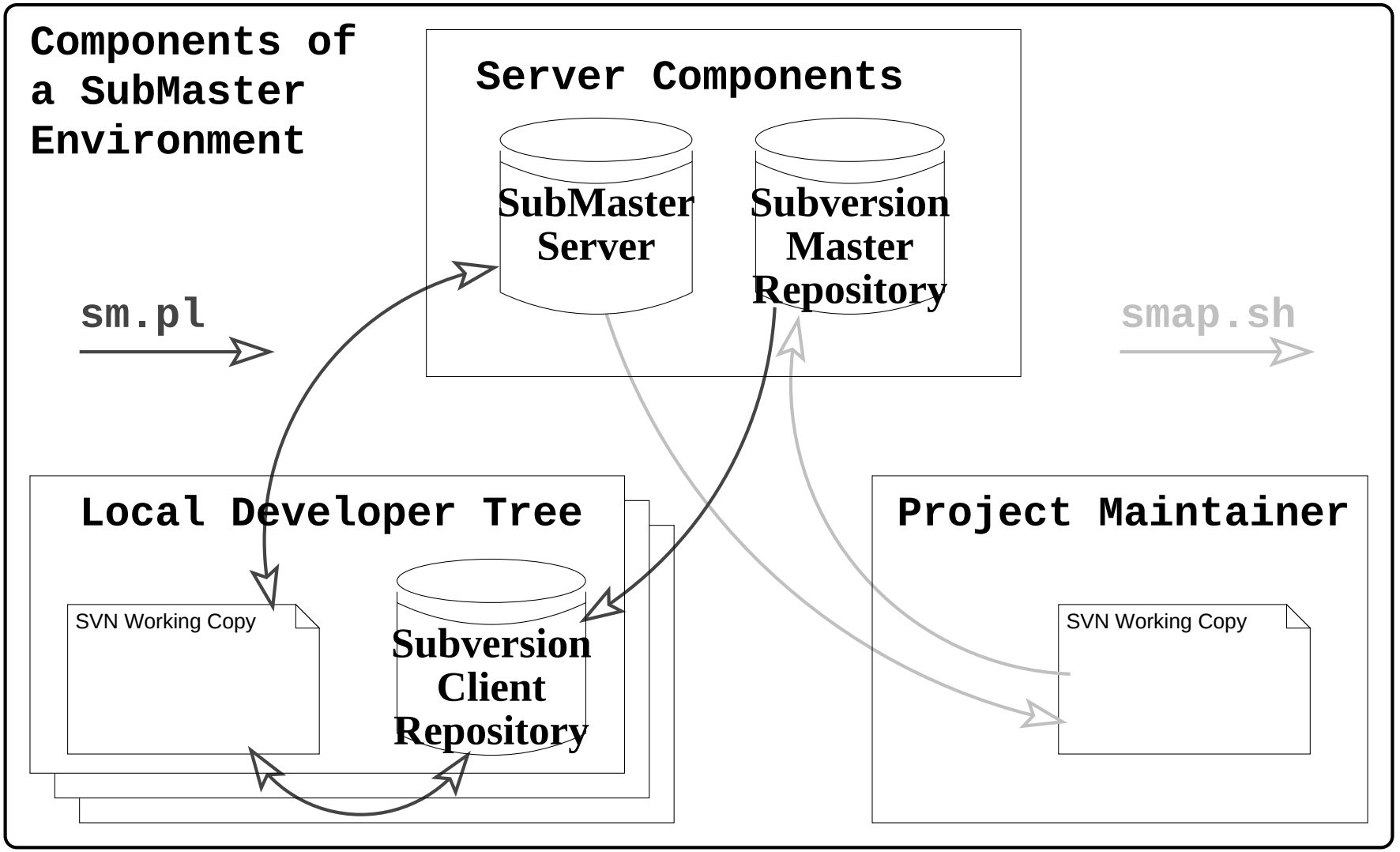
- SubMaster is a set of scripts based on Subversion targeting bazaar-style development
- Everyone has his own local Subversion repository
- SubMaster keeps the local repository in sync with the master repository preserving local changes
- SubMaster provides an infrastructure for sending patches, collecting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

LINBIT Information Technologies GmbH

What is SubMaster

- 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

Components



Using Subversion

Creating a repository

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

Creating a repository

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

LINBIT Information Technologies GmbH

Creating a repository

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

Creating a working copy

- Using the tool `svn` is very simmilar to using `cvs`.
- 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.

Creating a working copy

- Using the tool `svn` is very simmilar to using `cvs`.
- 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.

Creating a working copy

- Using the tool `svn` is very simmilar to using `cvs`.
- 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.

Creating a working copy

- Using the tool `svn` is very simmilar to using `cvs`.
- 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)

- 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`

LINBIT Information Technologies GmbH

Making changes (1)

- 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`

LINBIT Information Technologies GmbH

Making changes (1)

- 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)

- 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
    ```
    ```

Making changes (2)

- 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
    ```
    ```

Making changes (2)

- 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

- Commiting 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`
- 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 .
```

LINBIT Information Technologies GmbH

Committing changes

- Commiting 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`
- 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 .
```

Committing changes

- Commiting 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`
- 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 .
```

Committing changes

- Commiting 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`
- 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 .
```

Branches and tags

- 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
```

- Changes in the branch can be merged back using the `svn merge` command.
- A “tagged version” is also just a copy. It just never gets modified.

Branches and tags

- 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
```

- Changes in the branch can be merged back using the `svn merge` command.

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

Branches and tags

- 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
```

- Changes in the branch can be merged back using the `svn merge` command.
- A “tagged version” is also just a copy. It just never gets modified.

Branches and tags

- 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
```

- Changes in the branch can be merged back using the `svn merge` command.
- A “tagged version” is also just a copy. It just never gets modified.

Branches and tags

- 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
```

- Changes in the branch can be merged back using the `svn merge` command.

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

Getting Help (1)

- 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 (1)

- 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)

- 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/

Getting Help (2)

- 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/

Getting Help (2)

- 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/

The SubMaster Client

LINBIT Information Technologies GmbH

Overview

- 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`

LINBIT Information Technologies GmbH

Overview

- 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`

LINBIT Information Technologies GmbH

Overview

- 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

- 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 a local tree

- 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 a local tree

- 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>
  ```

LINBIT Information Technologies GmbH

Creating a local tree

- 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>
  ```

LINBIT Information Technologies GmbH

Creating patches (1)

- 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.

LINBIT Information Technologies GmbH

Creating patches (1)

- 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 (1)

- 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 (1)

- 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)

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

```
r44 | root | 2004-01-10 12:10:58 +0100 (Sat,
Added package oprofile.
------------------------------------------------
r51 | root | 2004-01-10 13:34:35 +0100 (Sat,
Oprofile:  don't run depmod.
------------------------------------------------
```

- 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.

Creating patches (2)

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

```
r44 | root | 2004-01-10 12:10:58 +0100 (Sat,
Added package oprofile.
------------------------------------------------
r51 | root | 2004-01-10 13:34:35 +0100 (Sat,
Oprofile:  don't run depmod.
------------------------------------------------
```

- 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.

Creating patches (2)

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

```
r44 | root | 2004-01-10 12:10:58 +0100 (Sat,
Added package oprofile.
------------------------------------------------
r51 | root | 2004-01-10 13:34:35 +0100 (Sat,
Oprofile:  don't run depmod.
------------------------------------------------
```

- 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

- Merging changes from master tree to local tree:
  - `sm sync`
  - manually resolve conflicts, if any
  - patches beeing 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`

Syncing with the main tree

- Merging changes from master tree to local tree:
  - `sm sync`
  - manually resolve conflicts, if any
  - patches beeing 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`

Syncing with the main tree

- Merging changes from master tree to local tree:
  - `sm sync`
  - manually resolve conflicts, if any
  - patches beeing 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`

WIP Archive

- 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`

WIP Archive

- 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`

WIP Archive

- 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`

WIP Archive

- 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`

WIP Archive

- 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`

WIP Archive

- 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)

- `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)

- `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

The SubMaster Server

Overview

- 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

Overview

- 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

Overview

- 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

Overview

- 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

Overview

- 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

Overview

- 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

Overview

- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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

Patch Properties

- A Unique Patch ID (such as "20040110203646150 18")
- 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)

- `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`
  empy file which marks a patch as open

Filesystem Layout (2)

- `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`
  eighter "Applied" or "Rejected" does not exist for open patches

The smap helper script

Overview

- 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: `smap [options] <patch-id>`

LINBIT Information Technologies GmbH

Overview

- 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: `smap [options] <patch-id>`

Overview

- 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: `smap [options] <patch-id>`

LINBIT Information Technologies GmbH

Overview

- 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: `smap [options] <patch-id>`

Overview

- 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: `smap [options] <patch-id>`

Options

- -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

Configuration

- 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"
```

Configuration

- 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"
```

Configuration

- 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"
```

Fixdiff

- Can be used to create patches for patches (e.g. so they apply cleanly)
- `smap 20040405094233279l3` (this fails)
- `fixdiff 20040405094233279l3.patch co package/base/linux24/lx_config.sh`
- Merge `package/base/linux24/lx_config.sh.rej`
- `fixdiff 20040405094233279l3.patch ci package/base/linux24/lx_config.sh`
- `--> fixdiff_20040405094233279l3.patch`

Fixdiff

- Can be used to create patches for patches (e.g. so they apply cleanly)
- `smap 20040405094233279l3` (this fails)
- `fixdiff 20040405094233279l3.patch co package/base/linux24/lx_config.sh`
- Merge `package/base/linux24/lx_config.sh.rej`
- `fixdiff 20040405094233279l3.patch ci package/base/linux24/lx_config.sh`
- `--> fixdiff_20040405094233279l3.patch`

Fixdiff

- Can be used to create patches for patches (e.g. so they apply cleanly)
- `smap 20040405094233279l3` (this fails)
- `fixdiff 20040405094233279l3.patch co package/base/linux24/lx_config.sh`
- Merge `package/base/linux24/lx_config.sh.rej`
- `fixdiff 20040405094233279l3.patch ci package/base/linux24/lx_config.sh`
- `--> fixdiff_20040405094233279l3.patch`

Fixdiff

- Can be used to create patches for patches (e.g. so they apply cleanly)
- `smap 20040405094233279l3` (this fails)
- `fixdiff 20040405094233279l3.patch co package/base/linux24/lx_config.sh`
- Merge `package/base/linux24/lx_config.sh.rej`
- `fixdiff 20040405094233279l3.patch ci package/base/linux24/lx_config.sh`
- `--> fixdiff_20040405094233279l3.patch`

Fixdiff

- Can be used to create patches for patches (e.g. so they apply cleanly)
- `smap 20040405094233279l3` (this fails)
- `fixdiff 20040405094233279l3.patch co package/base/linux24/lx_config.sh`
- Merge `package/base/linux24/lx_config.sh.rej`
- `fixdiff 20040405094233279l3.patch ci package/base/linux24/lx_config.sh`
- `--> fixdiff_20040405094233279l3.patch`

Fixdiff

- Can be used to create patches for patches (e.g. so they apply cleanly)
- `smap 20040405094233279l3` (this fails)
- `fixdiff 20040405094233279l3.patch co package/base/linux24/lx_config.sh`
- Merge `package/base/linux24/lx_config.sh.rej`
- `fixdiff 20040405094233279l3.patch ci package/base/linux24/lx_config.sh`
- `--> fixdiff_20040405094233279l3.patch`

SubMaster Action Scripts

Overview

- 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

Overview

- 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

Overview

- 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

Overview

- 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

Overview

- 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

Overview

- 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

- `./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`

URLs and References

Subversion and SubMaster

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

Related Projects

- 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/

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