The COOPY Toolbox

0.5.8

Generated by Doxygen 1.7.1

Fri Nov 4 2011 16:42:25

Contents

1 COOPY Manual

The COOPY project contains programs and libraries to help you:

- Describe the difference between two tables (see ssdiff).
- Apply the difference between two tables to a table (see sspatch).
- Merge tables that came from the same source but have been modified independently (see ssmerge). Two- and three-way merges supported.
- Maintain a repository of tables with multiple active contributors (see ssfossil and the coopy program).
- Supported table formats include CSV, sqlite, mysql, Excel, Access (read only).

1.1 Tutorials

- Contribute to data collection projects using ssdiff
- Pulling in a repository with Coopy
- Creating a new repository for Coopy
- Pushing out a repository with Coopy
- Forking a repository for Coopy
- Resolving conflicts with Coopy
- Using ssmerge with git (advanced)

1.2 Command line tools

- ssdiff
- sspatch
- ssmerge
- ssresolve
- ssformat
- ssrediff
- coopy

1.3 Reference material

- Diff format.
- Supported table formats.
- What are all these funny files in a Coopy directory?
- ssfossil reference.

And if COOPY isn't quite right for you?

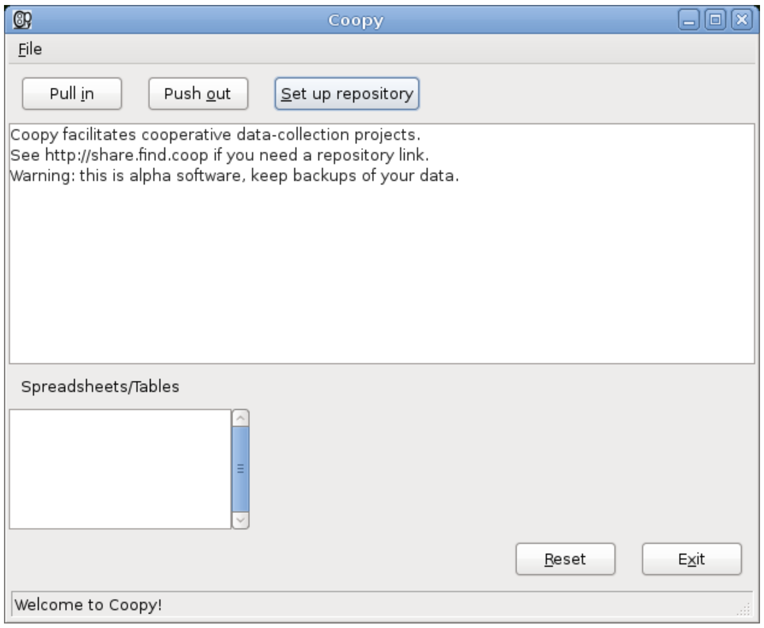
- Related projects.

Code in the COOPY project is all free and open source (licensed under the GPL; LGPL port in the works). The ssmerge and ssdiff programs currently make use of `libcsv`. The ssfossil program is a thin wrapper around `fossil`. The coopy program uses `wxWidgets`. Optionally, programs also make use of the spreadsheet library within `Gnumeric`.

2 Pulling in a repository with Coopy

The **coopy** program is a graphical interface for "pulling" new or updated repositories onto one's computer, and for "pushing" modified repositories out to a server. It is specialized for spreadsheets and databases. Advanced users who need maximum flexibility may wish to try to the ssfossil command-line tool rather than the coopy program. It is fine to use both on the same repository.

When you start the coopy program, you see something like this:



```
Coopy facilitates cooperative data-collection projects.
See http://share.find.coop if you need a repository link.
Warning: this is alpha software, keep backups of your data.
```

```
Welcome to Coopy!
```

Figure 1: Here's Coopy!

Here are the main parts of the interface:

- Buttons!
- A big text box, called the "activity log". When we start working with Coopy, messages will roll past in this box. This is a user interface designed by a computer scientist, sorry...
- A box labeled "Spreadsheets/Tables". A list of available databases appears here. Just now, there are none. We need to set up a repository first.

To do anything useful with Coopy, we need to get a repository. Click on the "Pull in" button to pull data in to your computer from the web, and you'll see:

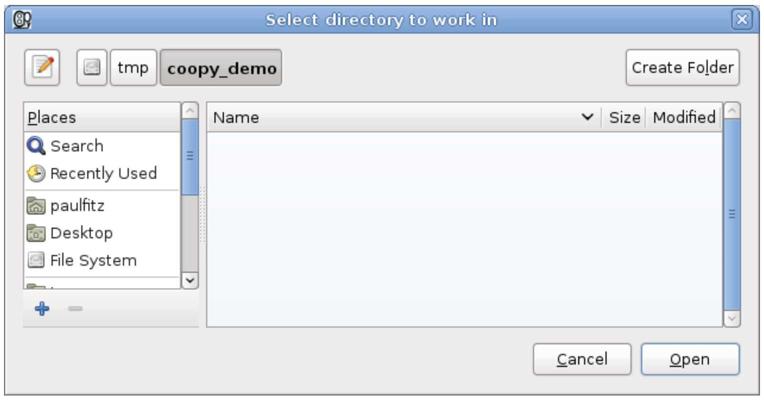
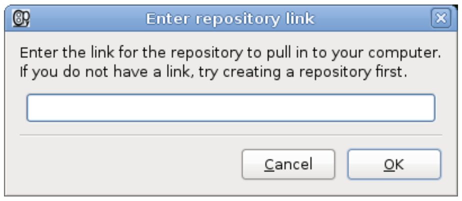


Figure 2: Select or create an empty directory.

What Coopy is looking for here is a fresh, clean, empty directory in which to store a bunch of repository files. It is a good idea to create a new folder for this purpose.

Once you've given Coopy an empty directory to fool around in, you're prompted for an address of a repository:



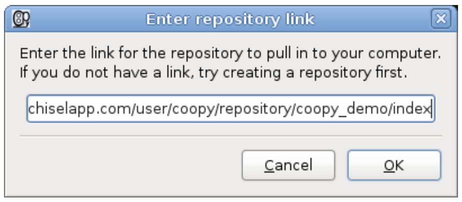
Enter the link for the repository to pull in to your computer.
If you do not have a link, try creating a repository first.

Figure 3: What to put here?

What the heck do we put here? Repositories live on the web, and have addresses like any webpage. We could make our own (see Creating a new repository for Coopy), but for now let's use one I've prepared earlier - this:

https://chiselapp.com/user/coopy/repository/coopy_-
demo/index

If you click on that link, you'll see something that has a little logo of a fossil. Don't bother poking around there yet, we'll come back to it later. For now, just paste in the link to Coopy:

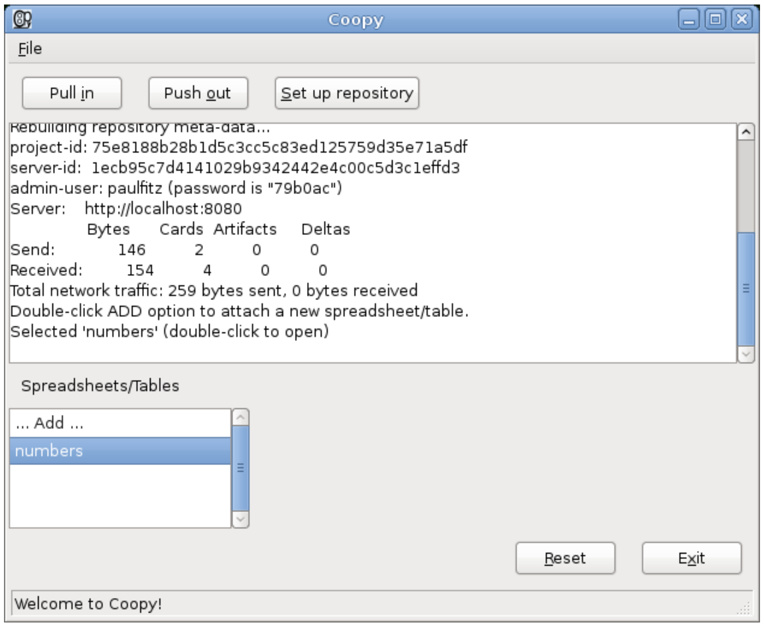


Enter the link for the repository to pull in to your computer.
If you do not have a link, try creating a repository first.

chiselapp.com/user/coopy/repository/coopy_demo/index

Figure 4: Fill in demo repository link.

Click OK, and Coopy chirps away merrily to itself, throwing messages into the activity log:

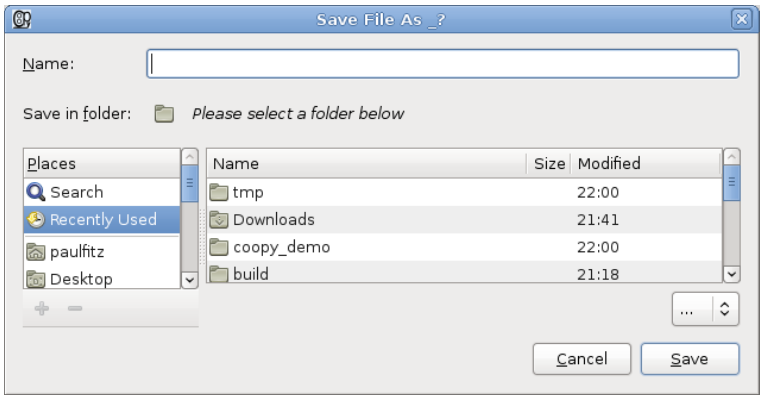


```
Rebuilding repository meta-data...
project-id: 75e8188b28b1d5c3cc5c83ed125759d35e71a5df
server-id:  1ecb95c7d4141029b9342442e4c00c5d3c1effd3
admin-user: paulfitz (password is "79b0ac")
Server:    http://localhost:8080
             Bytes      Cards  Artifacts     Deltas
Send:           146          2          0          0
Received:       154          4          0          0
Total network traffic: 259 bytes sent, 0 bytes received
Double-click ADD option to attach a new spreadsheet/table.
Selected 'numbers' (double-click to open)
```

Welcome to Coopy!

Figure 5: Our first repository has arrived!

No error messages, that's good. The "Spreadsheets/Tables" area has changed. It gives a list of databases in the repository. There is in fact just one, called "numbers". Double-click on it, and you're asked where you want to save the database, and in what format:



Name:

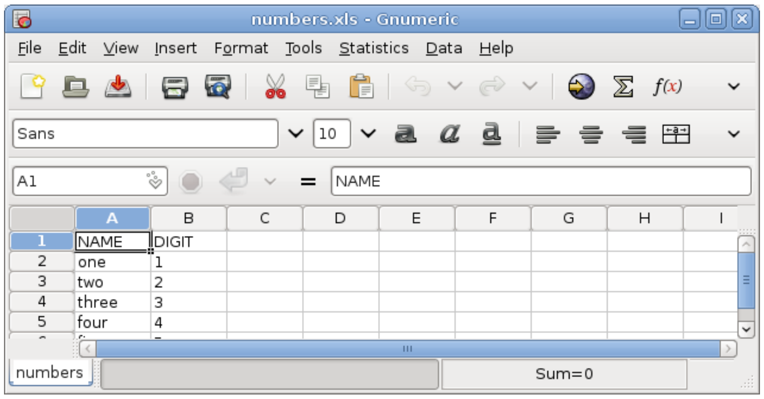
Save in folder: *Please select a folder below*

Places
Search
Recently Used
paulfitz
Desktop

Name	Size	Modified
tmp		22:00
Downloads		21:41
coopy_demo		22:00
build		21:18

Figure 6: Save database as...

You can choose to save the database in a number of formats: principally Excel, Sqlite, or CSV (for single-table databases). The choice is yours. For now, save the database as an ∗.xls file, and it will open in whatever you use to view spreadsheets on your computer (if you have nothing, install OpenOffice, Gnumeric, or something along those lines).



A1 = NAME

	A	B
1	NAME	DIGIT
2	one	1
3	two	2
4	three	3
5	four	4

Figure 7: The numbers spreadsheet.

What happens when we start making changes in the spreadsheet? That’s the topic of this tutorial: Pushing out a repository with Coopy.

3 Creating a new repository for Coopy

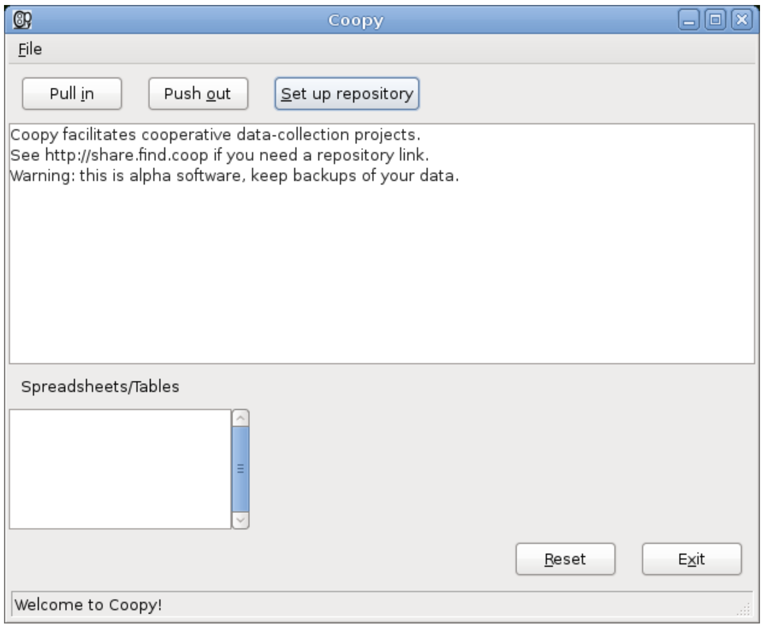
A repository is a collection of files. For Coopy, collections of files are managed by fossil, a distributed version control system. What does this mean?

- A revision history is kept for all files in the collection.
- The repository can be "cloned" on another computer or another location, in such a way that changes to the repository can be passed on easily.
- Clones of the repository can be changed by different people, and then have those changes merged intelligently.

It is possible to set up a repository using the fossil or ssfossil program. Alternatively, you can find or create a fossil repository online. Here are known hosting services for fossil repositories:

- `chiselapp.com`
- `share.find.coop`

You can find this list by starting coopy, then clicking on "Set up repository":



Coopy facilitates cooperative data-collection projects.
See http://share.find.coop if you need a repository link.
Warning: this is alpha software, keep backups of your data.

Welcome to Coopy!

Figure 8: Click on: Set up repository

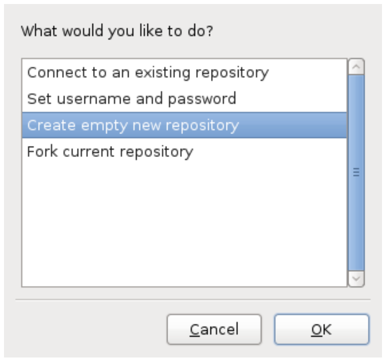

What would you like to do?
Connect to an existing repository
Set username and password
Create empty new repository
Fork current repository
Cancel
OK

Figure 9: Select: Create new repository

When we choose the new repository option, we are offered a list of hosts:

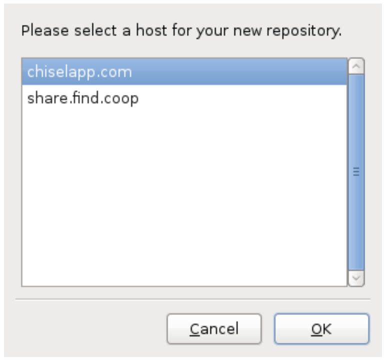
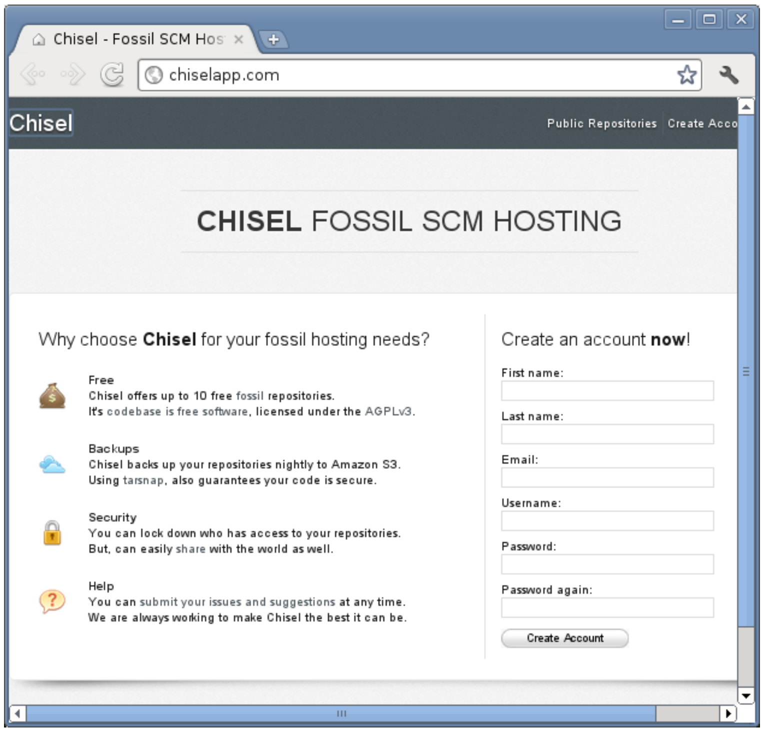


Figure 10: Select a host.

Choose a website to host your repository. You can also have your own website to host a fossil repository (see `self-hosting instructions`), you don't need to stick with this list. For now, let's go with `chiselapp.com`.


Chisel - Fossil SCM Hos

chiselapp.com

Chisel

Public Repositories Create Acco

CHISEL FOSSIL SCM HOSTING

Why choose **Chisel** for your fossil hosting needs?

Free
Chisel offers up to 10 free fossil repositories.
It's codebase is free software, licensed under the AGPLv3.

Backups
Chisel backs up your repositories nightly to Amazon S3.
Using tarsnap, also guarantees your code is secure.

Security
You can lock down who has access to your repositories.
But, can easily share with the world as well.

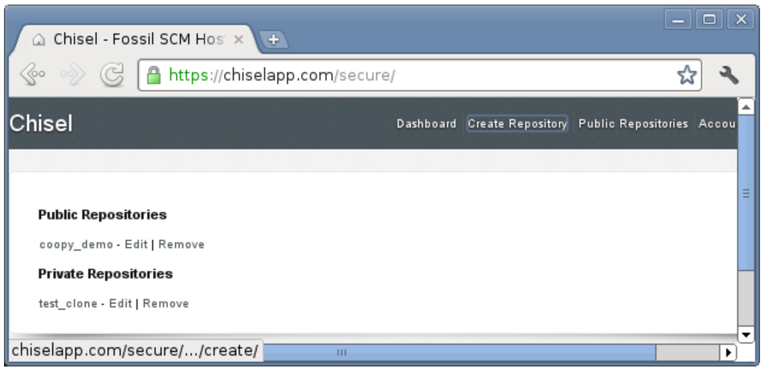
Help
You can submit your issues and suggestions at any time.
We are always working to make Chisel the best it can be.

Create an account **now**!

First name:	
Last name:	
Email:	
Username:	
Password:	
Password again:	

Figure 11: Chisel

After you register and log in, you'll see a dashboard that looks like this (except emptier):


Chisel - Fossil SCM Hos

https://chiselapp.com/secure/

Chisel

Public Repositories

coopy_demo - Edit | Remove

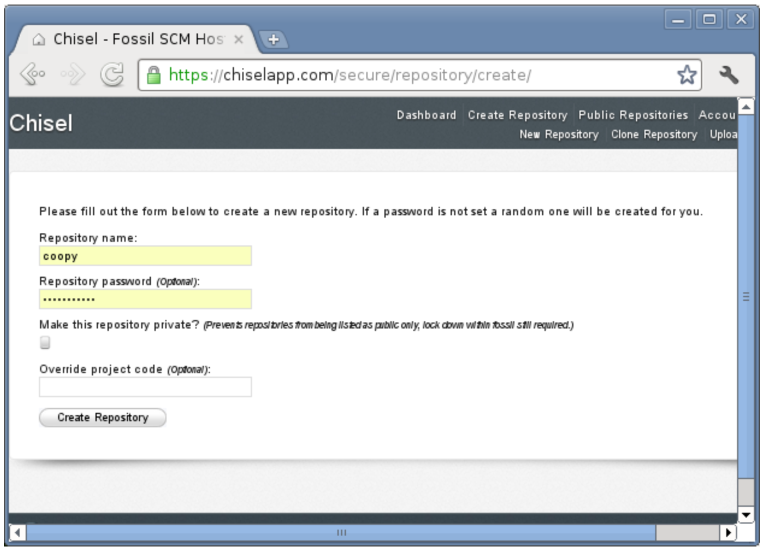
Private Repositories

test_clone - Edit | Remove

chiselapp.com/secure/.../create/

Figure 12: Chisel dashboard

To make a repository, click on "Create repository".



Please fill out the form below to create a new repository. If a password is not set a random one will be created for you.

Repository name:
coopy

Repository password *(Optional)*:
•••••••••••

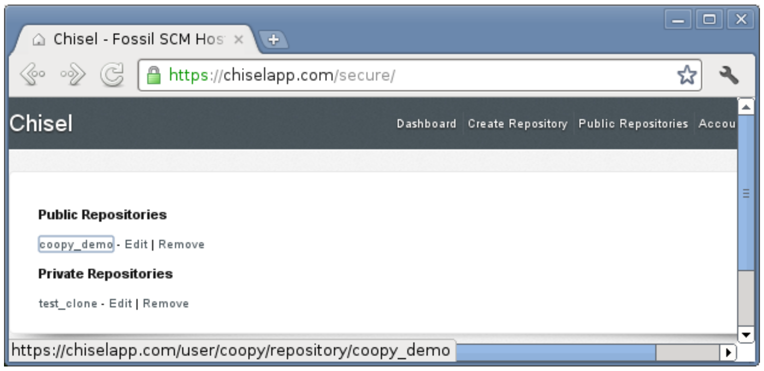
Make this repository private? *(Prevents repositories from being listed as public only, lock down within fossil still required.)*

Override project code *(Optional)*:

Create Repository

Figure 13: Create repository

Set the repository up as you like. The "Override project code" field is not needed when setting up a fresh, empty repository. Once you've set up a repository, on the dashboard find a link to it, like the one shown in a box here:



Public Repositories

coopy_demo - Edit | Remove

Private Repositories

test_clone - Edit | Remove

https://chiselapp.com/user/coopy/repository/coopy_demo

Figure 14: Find the repository link

When you click that link, you should see a page in a different style with a picture of a fossil. This is your repository, and the link shown in your browser is the "repository link" that Coopy needs.

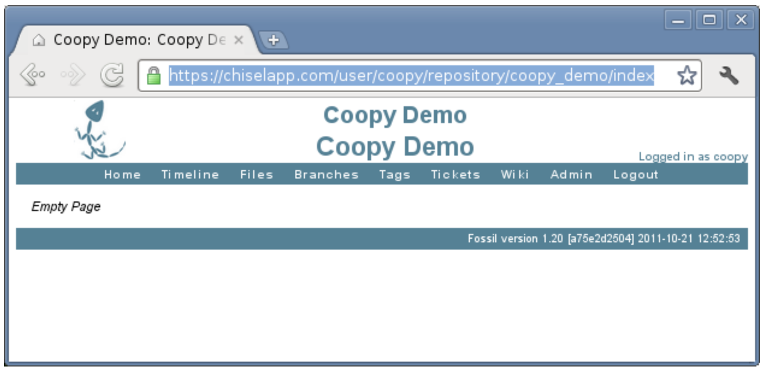

Coopy Demo: Coopy De
https://chiselapp.com/user/coopy/repository/coopy_demo/index
Coopy Demo
Coopy Demo
Logged in as coopy
Home Timeline Files Branches Tags Tickets Wiki Admin Logout
Empty Page
Fossil version 1.20 [a75e2d2504] 2011-10-21 12:52:53

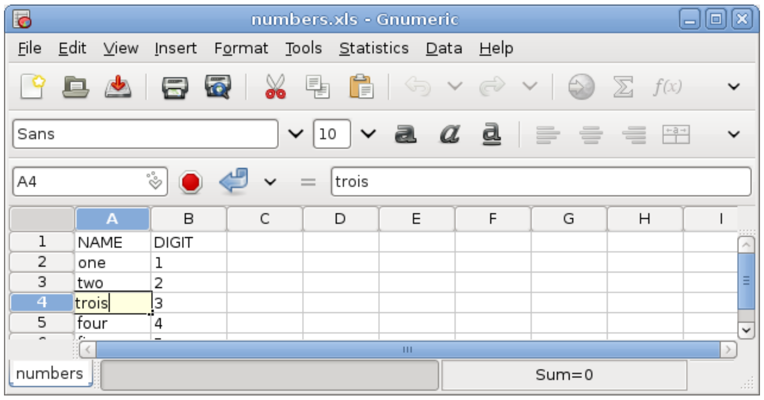
Figure 15: Fossil

You could now follow the instructions in this tutorial: Pulling in a repository with Coopy. Just replace the demo repository link with your own. You can also follow this tutorial: Pushing out a repository with Coopy, using your Chiselapp username, and the password you chose for the repository.

4 Pushing out a repository with Coopy

Do this tutorial first: Pulling in a repository with Coopy

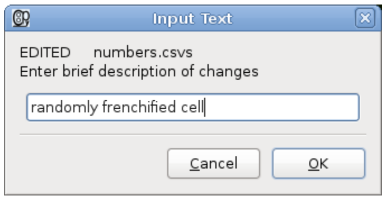
Now, change some number in your ∗.xls file, then save it.



	A	B
1	NAME	DIGIT
2	one	1
3	two	2
4	trois	3
5	four	4

Figure 16: Making a change...

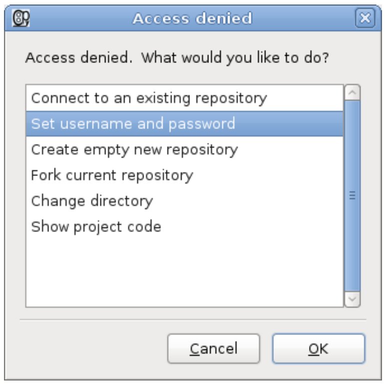
Now to send the updates to the repository, we click the "Push out" button. We're prompted to make a brief description of our changes (called a commit message). Write anything you like here. Typically this should be a brief message that will let a collaborator understand your change quickly.



EDITED numbers.csvs
Enter brief description of changes
randomly frenchified cell

Figure 17: Add a commit message.

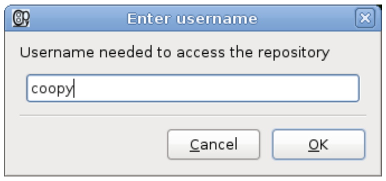
The next thing that is likely to happen is an access denied message. Repositories can be set up so that random folks can commit changes to them, but normally you need a username and password. Let's see how things work if you have a username/password (otherwise consider following: Forking a repository for Coopy).



Access denied. What would you like to do?
Connect to an existing repository
Set username and password
Create empty new repository
Fork current repository
Change directory
Show project code

Figure 18: Access denied.

Fill out a username and password:



Username needed to access the repository
coopy

Figure 19: Username.

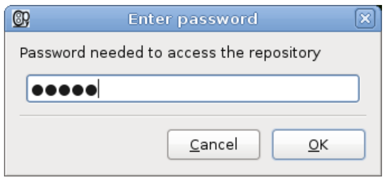
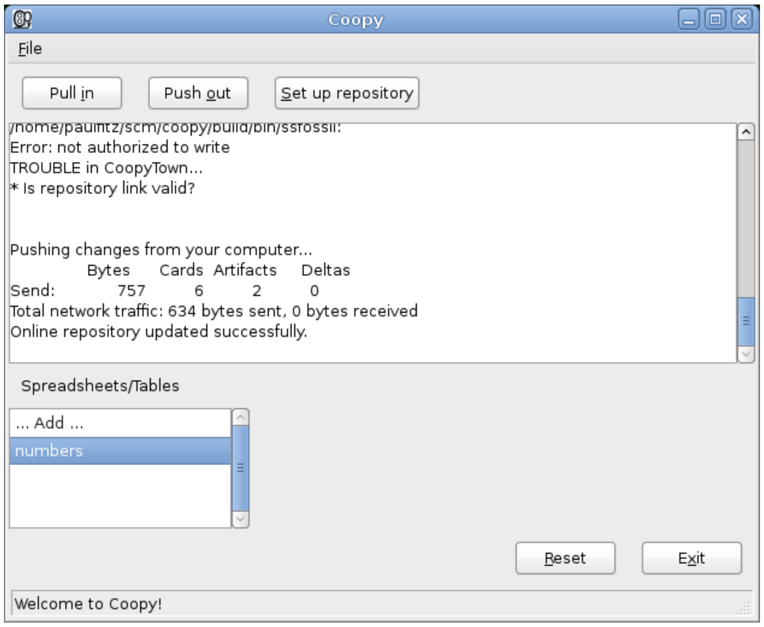


Figure 20: Password.

And we're done:



```
/home/paulfitz/scm/coopy/build/bin/ssfossil:
Error: not authorized to write
TROUBLE in CoopyTown...
* Is repository link valid?


Pushing changes from your computer...
             Bytes      Cards  Artifacts     Deltas
Send:           757          6          2          0
Total network traffic: 634 bytes sent, 0 bytes received
Online repository updated successfully.
```

Welcome to Coopy!

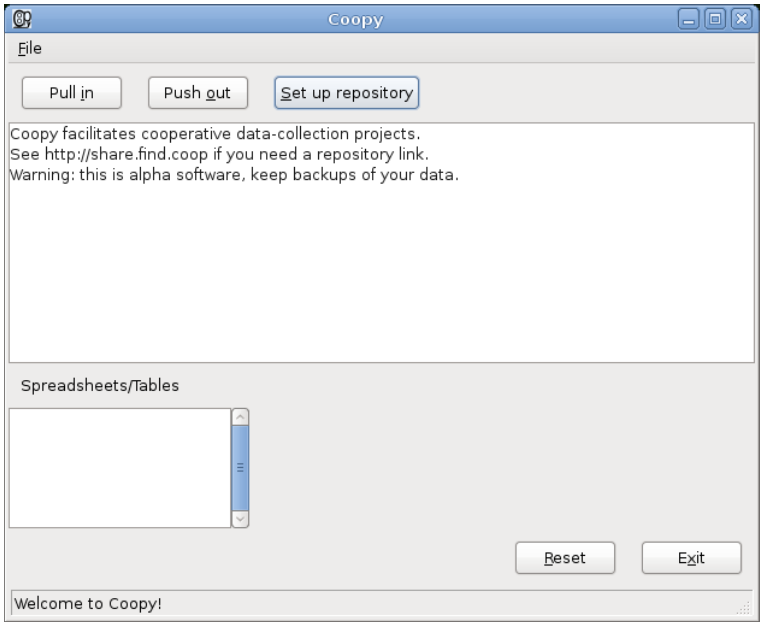
Figure 21: A successful push.

If you want to push out to a new repository, follow: Forking a repository for Coopy.

5 Forking a repository for Coopy

Suppose you pulled in the demo repository in Pulling in a repository with Coopy, then made modifications to it, but didn't want to (or couldn't) push those modifications back to the original repository. One option is to fork the repository and store it elsewhere.

The procedure is just like Creating a new repository for Coopy, except with one wrinkle: when making your new repository, you need to give it a "project code" that is identical to the one you pulled in. You can find that code by clicking on the "Set up repository" link:



Coopy facilitates cooperative data-collection projects.
See http://share.find.coop if you need a repository link.
Warning: this is alpha software, keep backups of your data.

Welcome to Coopy!

Figure 22: Click on: Set up repository

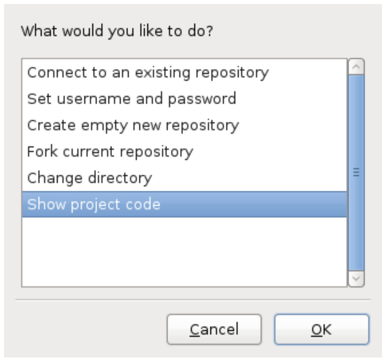
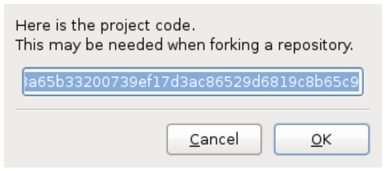

What would you like to do?
Connect to an existing repository
Set username and password
Create empty new repository
Fork current repository
Change directory
Show project code
Cancel
OK

Figure 23: Select: Show project code

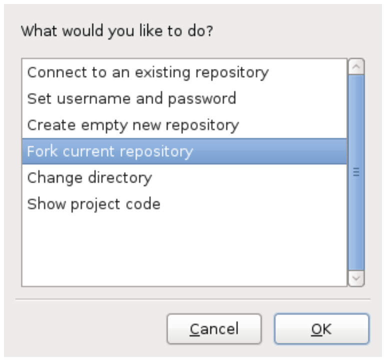
When we choose the project code option, Coopy gives us a big string of numbers and letters:


Here is the project code.
This may be needed when forking a repository.

a65b33200739ef17d3ac86529d6819c8b65c9

Figure 24: Project code

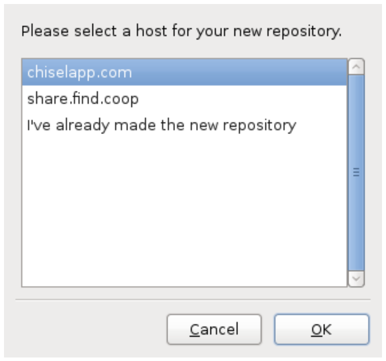
We will need the code a little later, don't worry about it for now, just be aware of where to find it. Clicking on the "Set up repository" link again and this time select the "fork repository" option:


What would you like to do?

Connect to an existing repository
Set username and password
Create empty new repository
Fork current repository
Change directory
Show project code

Figure 25: Fork repository option

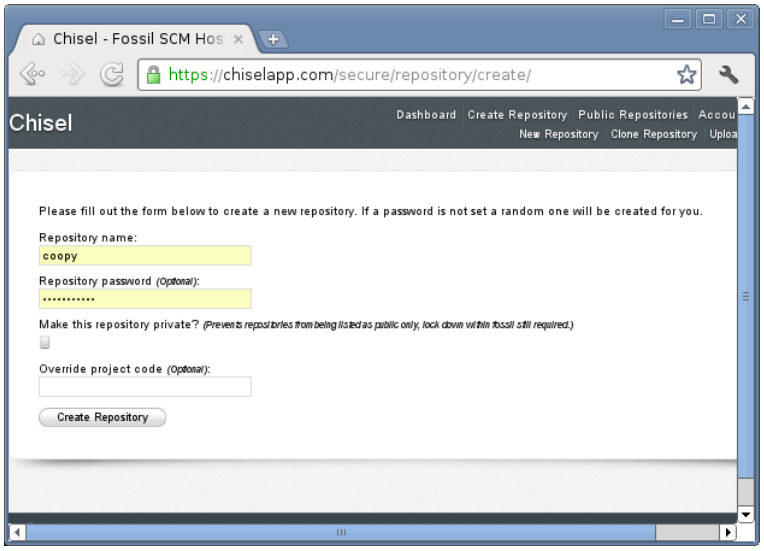
We get a list of possible hosts for our repository:


Please select a host for your new repository.

chiselapp.com
share.find.coop
I've already made the new repository

Figure 26: Fork host

Suppose we pick the first one. Then we follow the same sequence as in Creating a new repository for Coopy, up to this point:



Please fill out the form below to create a new repository. If a password is not set a random one will be created for you.

Repository name:
coopy

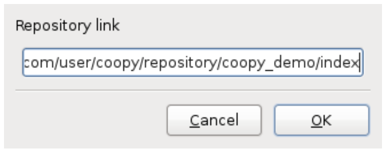
Repository password *(Optional)*:
•••••••••••

Make this repository private? *(Prevents repositories from being listed as public only, lock down within fossil still required.)*

Override project code *(Optional)*:

Figure 27: Create repository

Here we get the project code from Coopy and fill it in in the "Override project code" slot. Then, we get the repository link just like in Creating a new repository for Coopy, and give it to Coopy:



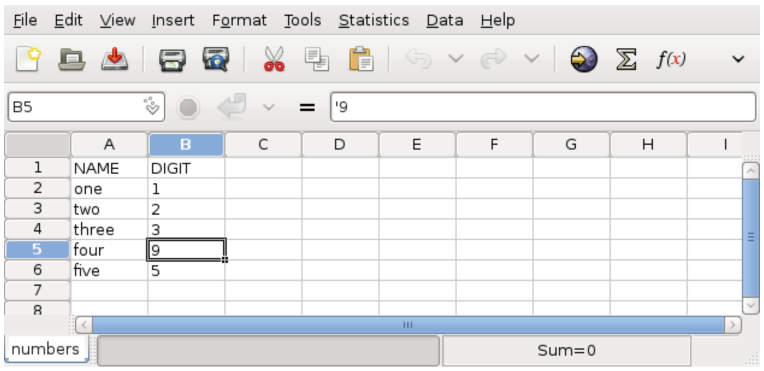
Repository link
com/user/coopy/repository/coopy_demo/index

Figure 28: Provide new repository link

Now, we continue on to provide a username and password to Coopy as in Pushing out a repository with Coopy. Done!

6 Resolving conflicts with Coopy

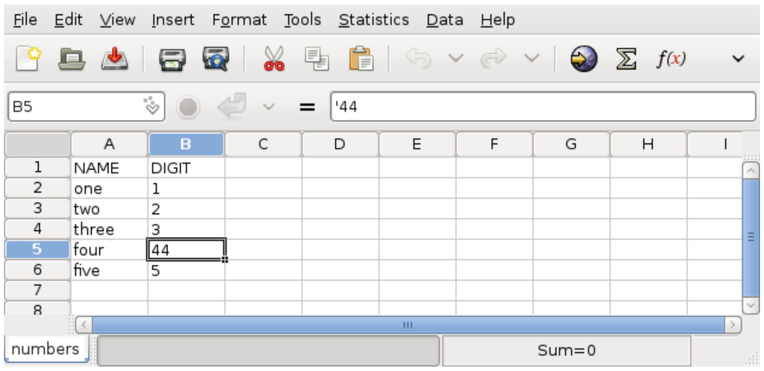
When working collaboratively, conflicts may occasionally happen. Suppose Alice and Bob both share the following table with Coopy:



	A	B	C	D	E	F	G	H	I
1	NAME	DIGIT							
2	one	1							
3	two	2							
4	three	3							
5	four	9							
6	five	5							
7									
8									

Figure 29: The starting point.

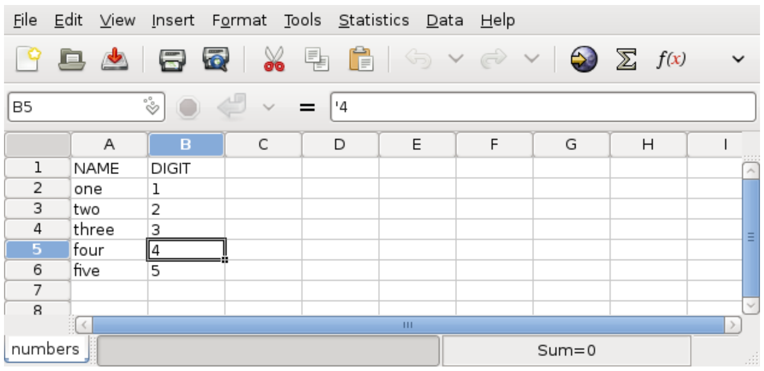
Both independently notice that the number "9" in this table is wrong, and fix it. Bob accidentally goofs:



	A	B	C	D	E	F	G	H	I
1	NAME	DIGIT							
2	one	1							
3	two	2							
4	three	3							
5	four	44							
6	five	5							
7									
8									

Figure 30: Bob’s fix

Alice gets it right:



'4

	A	B	C	D	E	F	G	H	I
1	NAME	DIGIT							
2	one	1							
3	two	2							
4	three	3							
5	four	4							
6	five	5							
7									
8									

Figure 31: Alice's fix

Suppose Bob pushes his "fix" first to their shared repository. When Alice tries to push her fix, Coopy gives this message:

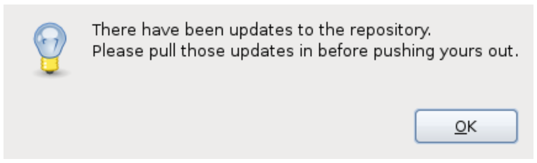
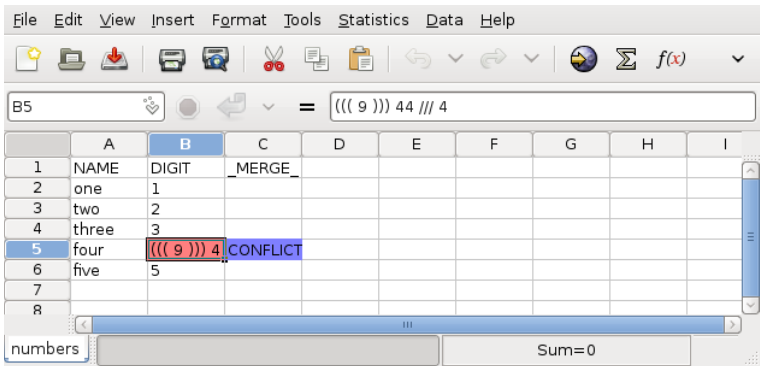

There have been updates to the repository.
Please pull those updates in before pushing yours out.

Figure 32: Pull needed.

Alice goes ahead and pulls. Compatible changes would get merged, but Coopy doesn't know what to do with the conflicting change. So it leaves Alice's table like this:



(((9))) 44 /// 4

	A	B	C	D	E	F	G	H	I
1	NAME	DIGIT	_MERGE_						
2	one	1							
3	two	2							
4	three	3							
5	four	(((9))) 4	CONFLICT						
6	five	5							
7									
8									

Figure 33: Conflicted table

Alice can recognize that "9" was replaced by either "4" (her choice) or "44" (Bob's choice). Using her magical human intelligence, she decides "4" is the right choice, and simply rewrites the conflicted cell, and deletes the "_MERGE_" column. Now she can push without trouble.

The Coopy toolbox has a ssresolve tool to speed up conflict resolving. Currently this works from the command-line only. Lots of graphical options should be integrated in the Coopy GUI soon.

7 Contribute to data collection projects using ssdiff

You can use ssdiff to summarize changes you've made between two versions of a table (or tables). Suppose you are a bridge geek, and have been working with this table of New York bridges compiled by a friend:

bridge	designer	length
Brooklyn	J. A. Roebling	1595
Williamsburg	D. Duck	1600
Queensborough	Palmer & Hornbostel	1182
Triborough	O. H. Ammann	1380,383
Bronx Whitestone	O. H. Ammann	2300
Throgs Neck	O. H. Ammann	1800
George Washington	O. H. Ammann	3500
Spamspan	S. Spamington	10000

That table has some problems (D. Duck? Spamington?). So you fix them.

bridge	designer	length
Brooklyn	J. A. Roebling	1595
Manhattan	G. Lindenthal	1470
Williamsburg	L. L. Buck	1600
Queensborough	Palmer & Hornbostel	1182
Triborough	O. H. Ammann	1380,383
Bronx Whitestone	O. H. Ammann	2300
Throgs Neck	O. H. Ammann	1800
George Washington	O. H. Ammann	3500

As a public-spirited type, you'd like to offer your fixes to your friend. But you don't want to spend much time at it. The easiest would be to just send your updated table. But that may be ignored, since there's no clear signal of what you've changed and whether it is worthwhile. So you run a quick diff and instead send this:

@	**bridge**	**designer**	**length**	
	Brooklyn	J. A. Roebling	1595	
+++	**Manhattan**	**G. Lindenthal**	**1470**	
->	Williamsburg	**D. Duck->L. L. Buck**	1600	
	Queensbor-ough	Palmer & Hornbostel	1182	
	Triborough	O. H. Ammann	1380,383	
...	...	...	...	
	Throgs Neck	O. H. Ammann	1800	
	George Washington	O. H. Ammann	3500	
---	Spamspan	S. Spamington	10000	

(produced with `ssdiff --format hilite --output patch.xls FILE1 FILE2`)

Or, if you and your friend are more unix-y, this:

```
* |bridge=Brooklyn|
+ |bridge:->Manhattan|designer:->G. Lindenthal|length:->1470|
= |bridge=Williamsburg|designer=D. Duck->L. L. Buck|
- |bridge=Spamspan|
```

(produced with `ssdiff FILE1 FILE2`)

Or, if you and your friend are more database-y, this:

```
INSERT INTO bridges (bridge, designer, length) VALUES ('Manhattan', 'G. Lindenthal', '1470');
UPDATE bridges SET designer='L. L. Buck' WHERE bridge='Williamsburg' AND designer='D. Duck' AND leng
DELETE FROM bridges WHERE bridge='Spamspan' AND designer='S. Spamington' AND length='10000';
```

(produced with `ssdiff --format sql --default-table bridges FILE1 FILE2`)

Now your friend can quickly see what changes you made, and update her/his own table either manually or using sspatch.

8 Using ssmerge with git (advanced)

You can use ssmerge to do smarter merges of tables stored in git.

- Add custom merge driver

- Add attributes file
- Worked example
- Troubleshooting
- Useful commands for testing

The basic steps are:

- Add "custom merge driver" lines to your .gitconfig file, to add coopy's ssmerge command as a merge option.
- Add format handlers to a .gitattributes file, to make sure the files you want are merged using ssmerge.

8.1 Add custom merge driver

Find or create a .gitconfig file in your home directory, OR find the file .git/config in a repository. Add to the end of this file lines like this:

```
[merge "coopy-csv"]
  name = coopy csv data merge
  driver = ssmerge --output dbi:csv::file=%A dbi:csv::file=%O dbi:csv::file=%A dbi:csv::file=%B
```

You'll need a variant of this stanza for each format you want ssmerge to handle. It is best to stick with text formats. The "dbi:csv::file=" part above is to force files to be in csv format (during merging their filename extensions are not known). If ssmerge is not in your path, replace "ssmerge" above with something like "/the/full/path/to/ssmerge".

8.2 Add attributes file

Find or create a .gitattributes file in the same directory as the files you want coopy to handle (there are other options for where to put this file, read the gitattributes documentation if you're interested). Add to the end of this file a line or lines like this:

```
*.csv merge=coopy-csv
```

The .gitattributes file can be placed under version control, so this only needs to get set up once. The custom merge driver step, on the other hand, needs to be set up for each collaborator.

8.3 Worked example

Let's make an empty git repository:

```
mkdir -p coopy_test/repo
cd coopy_test/repo
git init
```

Now, let's place a table in the repository. There are several options for the format we could use. For simplicity, let's start with a CSV file called "numbers.csv" with content like this:

```
NAME,DIGIT
one,1
two,2
thre,33
four,4
five,5
```

There are two intentional typos on the "thre" line. Add "numbers.csv" to the repository:

```
git add numbers.csv
git commit -m "add csv example"
```

Now, let's tell git to use a custom merge driver for .csv files. In the same directory as "numbers.csv", create a file called ".gitattributes" containing this:

```
*.csv merge=coopy-csv
```

Let's add this to the repository too:

```
git add .gitattributes
git commit -m "add coopy rule"
```

Now, at the end of $HOME/.gitconfig (create this file if it doesn't already exist), add on the following:

```
[merge "coopy-csv"]
  name = coopy csv data merge
  driver = ssmerge --format csv --output %A %O %A %B
```

Now let's set up a clone of this repository for testing:

```
cd ..   # should be in coopy_test directory now
git clone repo repo2
cd repo2
ls -a   # should see numbers.csv and .gitattributes
```

Good. Now, to test, we'll make two non-conflicting changes on the same row, and see if they get merged without a problem. Regular text-based merges will choke on this. So, in repo2, modify "numbers.csv" to make "thre" be "three", and commit:

```
git commit -m "fix three" numbers.csv
```

Now, in repo, modify "numbers.csv" to make "33" be "3", and commit:

```
cd ../repo
git commit -m "fix 3" numbers.csv
```

Now try merging:

```
git pull ../repo2
```

Happy result:

```
From ../repo2
 * branch            HEAD       -> FETCH_HEAD
Auto-merging numbers.csv
Merge made by recursive.
 numbers.csv |   12 ++++++------
 1 files changed, 6 insertions(+), 6 deletions(-)
```

And the contents of numbers.csv should be:

```
NAME,DIGIT
one,1
two,2
three,3
four,4
five,5
```

8.4 Troubleshooting

If for some reason git doesn't use the coopy merge rule, then something like the following message will be shown during merge:

```
remote: Counting objects: 5, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From ../repo2
 * branch            HEAD       -> FETCH_HEAD
Auto-merging numbers.csv
CONFLICT (content): Merge conflict in numbers.csv
Automatic merge failed; fix conflicts and then commit the result.
```

and numbers.csv will contain the following:

```
NAME,DIGIT
one,1
two,2
<<<<<<< HEAD
thre,3
=======
three,33
>>>>>>> b37613ebac50b552b4dd967c0f134930361c9070
four,4
five,5
```

This is the regular text merging algorithm. Undo the merge as follows:

```
git reset --hard HEAD   # remove any uncommitted changes
```

Then check:

- That the .gitconfig exists in your home directory and contains the coopy-csv rule described earlier.
- That there is a .gitattributes file in the same directory as numbers.csv, and that it has the contents described earlier.
- That the "ssmerge" command is in your path. If you run "ssmerge" you should see a help message.
- That "ssmerge" is a sufficiently recent version. Running "ssmerge --format csv" should not give an error message (compare against "ssformat --unknown-option csv")

8.5 Useful commands for testing

```
git reset --hard HEAD   # remove any uncommitted changes, such as a bad merge
git reset --hard HEAD^  # remove any uncommitted changes, and revert the last
                        # commit
git pull ../path        # merge in changes from another version of repo
git pull                # merge in changes from same repo as last time
```

9 ssdiff

Show the difference between two tables/databases/spreadsheets.

9.1 Usage

- ssdiff [options] FILE1 FILE2

9.2 Index

- Database/spreadsheet file formats
- Version

9.3 Option summary

- --apply
- --bid=COLUMN
- --default-table=TABLE
- --fixed-columns
- --format=FORMAT
- --head-trimmed
- --help
- --id=COLUMN
- --input-formats
- --named
- --omit-format-name
- --omit-sheet-name
- --output=OUTPUTFILE
- --parent=PARENT
- --patch-formats
- --table=TABLE
- --tail-trimmed
- --unordered

9.4 Option details

--apply

apply difference between FILE1 and FILE2 immediately to FILE1

--bid=COLUMN

boost a column (repeat option for multiple columns)

--default-table=TABLE

name to use when a table name is needed and not supplied

--fixed-columns

ignore new or removed columns

--format=FORMAT

set difference format for output

--head-trimmed

ignore rows removed at the beginning of a table (such as a log file)

--help

show how to use this program

--id=COLUMN

set primary key (repeat option for multi-column key)

--input-formats

list supported input database formats

--named

trust names of columns, omitting checks for column renames

--omit-format-name

omit any version-dependent header from diff

--omit-sheet-name

omit any sheet/table name from diff

--output=OUTPUTFILE

direct output to this file (default is standard output)

--parent=PARENT

use named workbook/database as common ancestor in difference calculations

--patch-formats

list supported patch formats

--table=TABLE

operate on a single named table of a workbook/database

--tail-trimmed

ignore rows removed at the end of a table (such as a log file)

--unordered

treat order of rows as unimportant

9.5 Examples

You can generate test file(s) for the examples that follow:

```
ssdiff --test-file numbers.csv
ssdiff --test-file numbers.sqlite
ssdiff --test-file numbers_buggy.csv
ssdiff --test-file numbers_buggy.sqlite
```

9.5.1 Example 1

```
ssdiff numbers_buggy.csv numbers.csv
```

Compare two tables. Output goes to standard output.

9.5.2 Example 2

```
ssdiff --unordered numbers_buggy.csv numbers.csv
```

Compare two tables, neglecting row order.

9.5.3 Example 3

```
ssdiff --format sql numbers_buggy.sqlite numbers.sqlite
```

Compare two databases, with output in SQL format.

9.5.4 Example 4

```
ssdiff --format hilite --output review.csv numbers_buggy.csv numbers.csv
```

Generate tabular diff for eyeballing. If ssdiff is compiled with gnumeric support, and output format is ∗.xls, color highlighting is added.

9.6 Patch formats

- **tdiff**: *[default]* vaguely similar to a standard unix diff
- **csv**: csv-compatible diff format
- **hilite**: colorful spreadsheet format
- **index**: tabular output showing relationship between rows and columns

- **raw**: verbose diff format for debugging
- **review**: spreadsheet diff format suitable for quickly accepting or rejecting changes
- **sql**: SQL format (data diffs only)

9.7 Database/spreadsheet file formats

CSV: plain-text delimiter-separated family of formats

- **.csv**: Comma-separated values
- **.tsv**: Tab-separated values
- **.ssv**: Semicolon-separated values
- **.json**: {
  "type": "csv",
  "file": "fname.dsv",
  "delimiter": "|"
  }

SQLITE: file-based database

- **.sqlite**: Sqlite database file
- **.json**: {
  "type": "sqlite",
  "file": "fname.db"
  }
- **dbi:sqlite:fname.db** (Force sqlite interpretation)

SQLITEXT: sqlite-format sql dump

- **.sqlitext**: SQL dump of Sqlite database
- **.json**: {
  "type": "sqlitext",
  "file": "fname.sql"
  }
- **dbi:sqlitext:fname.sql** (Force sqlitext interpretation)

JSONBOOK: Spreadsheet formats in json

- **.jsonbook**: Json spreadsheet

GNUMERIC: Spreadsheet formats (via gnumeric)

- **.xls**: Excel spreadsheet
- **.xlsx**: Excel spreadsheet
- **.gnumeric**: Gnumeric spreadsheet
- **.json**: {
  "type": "gnumeric",
  "file": "fname.sheet"
  }
- **dbi:gnumeric:fname.sheet** (Force Gnumeric interpretation)

MDB: Access database format (via Mdbtools, READ-ONLY)

- **.mdb**: Access database file
- **.json**: {
  "type": "access",
  "file": "fname.db"
  }
- **dbi:access:fname.db** (Force Access interpretation)

MYSQL: database connector

- **.json**: {
  "type": "mysql",
  "database": "db_name",
  "host": "localhost",
  "port": "1111",
  "username": "root",
  "password": "****"
  }
- **dbi:mysql:database_name** (Use default port, username, etc)
- **dbi:mysql:database_name:username=USER:password=PASS**
- **dbi:mysql:database_name:host=HOST:port=PORT**

9.8 Version

ssdiff version 0.5.8

10 sspatch

Modify a table/database/spreadsheet to integrate the changes described in a pre-computed difference.

10.1 Usage

- sspatch [options] DATAFILE PATCHFILE
- sspatch [options] --cmd PATCHSTRING DATAFILE

10.2 Index

10.3 Option summary

- --cmd=CMD
- --default-table=TABLE
- --help
- --inplace
- --input-formats
- --output=OUTPUTFILE
- --patch-formats
- --table=TABLE

10.4 Option details

--cmd=CMD

specify a patch (in tdiff format) with a string rather than as a file, useful to make a quick change to a table that does not merit a full patch file

--default-table=TABLE

name to use when a table name is needed and not supplied

--help

show how to use this program

--inplace

if modifications are made, make them in place without a copy

--input-formats

list supported input database formats

--output=OUTPUTFILE

direct output to this file (default is standard output)

--patch-formats

list supported patch formats

--table=TABLE

operate on a single named table of a workbook/database

10.5 Examples

You can generate test file(s) for the examples that follow:

```
sspatch --test-file numbers.csv
sspatch --test-file numbers_buggy.csv
sspatch --test-file numbers_buggy.sqlite
```

10.5.1 Example 1

```
sspatch numbers_buggy.csv numbers_patch.tdiff
```

Apply a patch to a table. Output goes to standard output. Input file is untouched.

10.5.2 Example 2

```
sspatch --inplace numbers_buggy.csv numbers_patch.tdiff
```

Apply a patch to a table. Input file is modified.

10.5.3 Example 3

```
sspatch --tmp tmp.sqlite numbers_buggy.sqlite numbers_patch.tdiff
```

Apply a patch to a sqlite database. Input file is not modified. Space for a temporary database is needed to do this. If not supplied, sspatch will ask for it.

10.5.4 Example 4

```
sspatch - numbers_patch.tdiff < numbers_buggy.csv
```

Apply a patch to a table read from standard input.

10.5.5 Example 5

```
sspatch numbers_buggy.csv - < numbers_patch.tdiff
```

Apply a patch read from standard input.

10.5.6 Example 6

```
sspatch --cmd "+ |two|2|" numbers_buggy.csv
```

Add a new row to a table.

10.5.7 Example 7

```
sspatch --cmd "- |NAME=four|" numbers.csv
```

Remove a row from a table

10.5.8 Example 8

```
sspatch --cmd "= |NAME=four|DIGIT:*->4|" numbers_buggy.csv
```

Change the DIGIT column on a row with NAME=four.

10.6 Patch formats

- **tdiff**: *[default]* vaguely similar to a standard unix diff
- **csv**: csv-compatible diff format
- **hilite**: colorful spreadsheet format
- **index**: tabular output showing relationship between rows and columns
- **raw**: verbose diff format for debugging
- **review**: spreadsheet diff format suitable for quickly accepting or rejecting changes
- **sql**: SQL format (data diffs only)

10.7 Database/spreadsheet file formats

CSV: plain-text delimiter-separated family of formats

- **.csv**: Comma-separated values
- **.tsv**: Tab-separated values
- **.ssv**: Semicolon-separated values
- **.json**: {
  "type": "csv",
  "file": "fname.dsv",
  "delimiter": "|"
  }

SQLITE: file-based database

- **.sqlite**: Sqlite database file
- **.json**: {
  "type": "sqlite",
  "file": "fname.db"
  }
- **dbi:sqlite:fname.db** (Force sqlite interpretation)

SQLITEXT: sqlite-format sql dump

- **.sqlitext**: SQL dump of Sqlite database
- **.json**: {
  "type": "sqlitext",
  "file": "fname.sql"
  }
- **dbi:sqlitext:fname.sql** (Force sqlitext interpretation)

JSONBOOK: Spreadsheet formats in json

- **.jsonbook**: Json spreadsheet

GNUMERIC: Spreadsheet formats (via gnumeric)

- **.xls**: Excel spreadsheet
- **.xlsx**: Excel spreadsheet
- **.gnumeric**: Gnumeric spreadsheet
- **.json**: {
  "type": "gnumeric",
  "file": "fname.sheet"
  }
- **dbi:gnumeric:fname.sheet** (Force Gnumeric interpretation)

MDB: Access database format (via Mdbtools, READ-ONLY)

- **.mdb**: Access database file
- **.json**: {
  "type": "access",
  "file": "fname.db"
  }
- **dbi:access:fname.db** (Force Access interpretation)

MYSQL: database connector

- **.json**: {
  "type": "mysql",
  "database": "db_name",
  "host": "localhost",
  "port": "1111",
  "username": "root",
  "password": "∗∗∗∗"
  }
- **dbi:mysql:database_name** (Use default port, username, etc)
- **dbi:mysql:database_name:username=USER:password=PASS**
- **dbi:mysql:database_name:host=HOST:port=PORT**

10.8 Version

sspatch version 0.5.8

11 ssmerge

Merge table/database/spreadsheets. The first file must be a common ancestor of the remaining two.

11.1 Usage

- ssmerge [options] FILE1 FILE2 FILE3

11.2 Index

11.3 Option summary

- --bid=COLUMN
- --default-table=TABLE
- --fixed-columns
- --head-trimmed
- --help
- --id=COLUMN
- --inplace
- --input-formats
- --named
- --output=OUTPUTFILE
- --patch-formats
- --table=TABLE
- --tail-trimmed
- --unordered

11.4 Option details

--bid=COLUMN

boost a column (repeat option for multiple columns)

--default-table=TABLE

name to use when a table name is needed and not supplied

--fixed-columns

ignore new or removed columns

--head-trimmed

ignore rows removed at the beginning of a table (such as a log file)

--help

show how to use this program

--id=COLUMN

set primary key (repeat option for multi-column key)

--inplace

if modifications are made, make them in place without a copy

--input-formats

list supported input database formats

--named

trust names of columns, omitting checks for column renames

--output=OUTPUTFILE

direct output to this file (default is standard output)

--patch-formats

list supported patch formats

--table=TABLE

operate on a single named table of a workbook/database

--tail-trimmed

ignore rows removed at the end of a table (such as a log file)

--unordered

treat order of rows as unimportant

11.5 Examples

You can generate test file(s) for the examples that follow:

```
ssmerge --test-file numbers.csv
ssmerge --test-file numbers_buggy.csv
ssmerge --test-file numbers_buggy_add.csv
ssmerge --test-file numbers_conflict.csv
```

11.5.1 Example 1

```
ssmerge numbers_buggy.csv numbers.csv numbers_buggy_add.csv
```

Merge two CSV tables (numbers.csv and numbers_buggy_add.csv) with a common ancestor (numbers_buggy.csv).

11.5.2 Example 2

```
ssmerge --theirs numbers_buggy.csv numbers.csv numbers_conflict.csv
```

Merge numbers.csv and numbers_conflict.csv (with common ancestor numbers_-buggy.csv), deferring to numbers_conflict.csv in the case of conflict.

11.5.3 Example 3

```
ssmerge --ours numbers_buggy.csv numbers.csv numbers_conflict.csv
```

Merge numbers.csv and numbers_conflict.csv (with common ancestor numbers_-buggy.csv), deferring to numbers.csv in the case of conflict.

11.5.4 Example 4

```
ssmerge --inplace --theirs numbers_buggy.csv numbers.csv numbers_conflict.csv
```

Merge directly into numbers.csv. Without --inplace, output goes to standard output.

11.6 Patch formats

- **tdiff**: *[default]* vaguely similar to a standard unix diff
- **csv**: csv-compatible diff format
- **hilite**: colorful spreadsheet format
- **index**: tabular output showing relationship between rows and columns
- **raw**: verbose diff format for debugging
- **review**: spreadsheet diff format suitable for quickly accepting or rejecting changes
- **sql**: SQL format (data diffs only)

11.7 Database/spreadsheet file formats

CSV: plain-text delimiter-separated family of formats

- **.csv**: Comma-separated values
- **.tsv**: Tab-separated values
- **.ssv**: Semicolon-separated values

- **.json**: {
  "type": "csv",
  "file": "fname.dsv",
  "delimiter": "|"
  }

SQLITE: file-based database

- **.sqlite**: Sqlite database file
- **.json**: {
  "type": "sqlite",
  "file": "fname.db"
  }
- **dbi:sqlite:fname.db** (Force sqlite interpretation)

SQLITEXT: sqlite-format sql dump

- **.sqlitext**: SQL dump of Sqlite database
- **.json**: {
  "type": "sqlitext",
  "file": "fname.sql"
  }
- **dbi:sqlitext:fname.sql** (Force sqlitext interpretation)

JSONBOOK: Spreadsheet formats in json

- **.jsonbook**: Json spreadsheet

GNUMERIC: Spreadsheet formats (via gnumeric)

- **.xls**: Excel spreadsheet
- **.xlsx**: Excel spreadsheet
- **.gnumeric**: Gnumeric spreadsheet
- **.json**: {
  "type": "gnumeric",
  "file": "fname.sheet"
  }

- **dbi:gnumeric:fname.sheet** (Force Gnumeric interpretation)

MDB: Access database format (via Mdbtools, READ-ONLY)

- **.mdb**: Access database file
- **.json**: {
  "type": "access",
  "file": "fname.db"
  }
- **dbi:access:fname.db** (Force Access interpretation)

MYSQL: database connector

- **.json**: {
  "type": "mysql",
  "database": "db_name",
  "host": "localhost",
  "port": "1111",
  "username": "root",
  "password": "****"
  }
- **dbi:mysql:database_name** (Use default port, username, etc)
- **dbi:mysql:database_name:username=USER:password=PASS**
- **dbi:mysql:database_name:host=HOST:port=PORT**

11.8 Version

ssmerge version 0.5.8

12 ssresolve

Resolve a file with conflicts from ssmerge.

12.1 Usage

- ssresolve [options] FILE

12.2 Index

12.3 Option summary

- --default-table=TABLE
- --dry-run
- --help
- --neither
- --ours
- --theirs

12.4 Option details

--default-table=TABLE

name to use when a table name is needed and not supplied

--dry-run

make no changes, just describe what would happen

--help

show how to use this program

--neither

in case of conflict use cell value from common ancestor

--ours

in case of conflict use cell value that was the local choice

--theirs

in case of conflict use cell value that wasn't the local choice

12.5 Examples

You can generate test file(s) for the examples that follow:

```
ssresolve --test-file numbers.csv
ssresolve --test-file numbers_buggy.csv
ssresolve --test-file numbers_conflict.csv
```

12.5.1 Example 1

```
ssresolve numbers_muddle.csv
```

Check if file is resolved.

12.5.2 Example 2

```
ssresolve --ours numbers_muddle.csv
```

Resolve conflicts in favor of local/left values.

12.5.3 Example 3

```
ssresolve --theirs numbers_muddle.csv
```

Resolve conflicts in favor of remote/right values.

12.5.4 Example 4

```
ssresolve --neither numbers_muddle.csv
```

Resolve conflicts in favor of ancestral values.

12.6 Patch formats

- **tdiff**: *[default]* vaguely similar to a standard unix diff
- **csv**: csv-compatible diff format
- **hilite**: colorful spreadsheet format
- **index**: tabular output showing relationship between rows and columns
- **raw**: verbose diff format for debugging

- **review**: spreadsheet diff format suitable for quickly accepting or rejecting changes
- **sql**: SQL format (data diffs only)

12.7 Database/spreadsheet file formats

CSV: plain-text delimiter-separated family of formats

- **.csv**: Comma-separated values
- **.tsv**: Tab-separated values
- **.ssv**: Semicolon-separated values
- **.json**: {
  "type": "csv",
  "file": "fname.dsv",
  "delimiter": "|"
  }

SQLITE: file-based database

- **.sqlite**: Sqlite database file
- **.json**: {
  "type": "sqlite",
  "file": "fname.db"
  }
- **dbi:sqlite:fname.db** (Force sqlite interpretation)

SQLITEXT: sqlite-format sql dump

- **.sqlitext**: SQL dump of Sqlite database
- **.json**: {
  "type": "sqlitext",
  "file": "fname.sql"
  }
- **dbi:sqlitext:fname.sql** (Force sqlitext interpretation)

JSONBOOK: Spreadsheet formats in json

- **.jsonbook**: Json spreadsheet

GNUMERIC: Spreadsheet formats (via gnumeric)

- **.xls**: Excel spreadsheet
- **.xlsx**: Excel spreadsheet
- **.gnumeric**: Gnumeric spreadsheet
- **.json**: {
  "type": "gnumeric",
  "file": "fname.sheet"
  }
- **dbi:gnumeric:fname.sheet** (Force Gnumeric interpretation)

MDB: Access database format (via Mdbtools, READ-ONLY)

- **.mdb**: Access database file
- **.json**: {
  "type": "access",
  "file": "fname.db"
  }
- **dbi:access:fname.db** (Force Access interpretation)

MYSQL: database connector

- **.json**: {
  "type": "mysql",
  "database": "db_name",
  "host": "localhost",
  "port": "1111",
  "username": "root",
  "password": "****"
  }
- **dbi:mysql:database_name** (Use default port, username, etc)
- **dbi:mysql:database_name:username=USER:password=PASS**
- **dbi:mysql:database_name:host=HOST:port=PORT**

12.8 Version

ssresolve version 0.5.8

13 ssformat

Reformat tables/databases/spreadsheets.

13.1 Usage

- ssformat [options] FILE
- ssformat [options] FILE1 FILE2

13.2 Index

13.3 Option summary

- --default-table=TABLE
- --header
- --help
- --index
- --input-formats
- --table=TABLE

13.4 Option details

--default-table=TABLE

name to use when a table name is needed and not supplied

--header

extract column names only

--help

show how to use this program

--index

extract content of key columns only

--input-formats

list supported input database formats

--table=TABLE

operate on a single named table of a workbook/database

13.5 Examples

You can generate test file(s) for the examples that follow:

```
ssformat --test-file numbers.csv
ssformat --test-file numbers.sqlite
```

13.5.1 Example 1

```
ssformat numbers.csv numbers_converted.sqlite
```

Convert CSV format table to an Sqlite database table.

13.5.2 Example 2

```
ssformat numbers.sqlite numbers_converted.csv
```

Convert Sqlite database table to a CSV format table.

13.5.3 Example 3

```
ssformat numbers.sqlite -
```

Display contents of an Sqlite database table.

13.6 Patch formats

- **tdiff**: *[default]* vaguely similar to a standard unix diff
- **csv**: csv-compatible diff format
- **hilite**: colorful spreadsheet format
- **index**: tabular output showing relationship between rows and columns
- **raw**: verbose diff format for debugging
- **review**: spreadsheet diff format suitable for quickly accepting or rejecting changes
- **sql**: SQL format (data diffs only)

13.7 Database/spreadsheet file formats

CSV: plain-text delimiter-separated family of formats

- **.csv**: Comma-separated values
- **.tsv**: Tab-separated values
- **.ssv**: Semicolon-separated values
- **.json**: {
  "type": "csv",
  "file": "fname.dsv",
  "delimiter": "|"
  }

SQLITE: file-based database

- **.sqlite**: Sqlite database file
- **.json**: {
  "type": "sqlite",
  "file": "fname.db"
  }
- **dbi:sqlite:fname.db** (Force sqlite interpretation)

SQLITEXT: sqlite-format sql dump

- **.sqlitext**: SQL dump of Sqlite database
- **.json**: {
  "type": "sqlitext",
  "file": "fname.sql"
  }
- **dbi:sqlitext:fname.sql** (Force sqlitext interpretation)

JSONBOOK: Spreadsheet formats in json

- **.jsonbook**: Json spreadsheet

GNUMERIC: Spreadsheet formats (via gnumeric)

- **.xls**: Excel spreadsheet
- **.xlsx**: Excel spreadsheet
- **.gnumeric**: Gnumeric spreadsheet
- **.json**: {
  "type": "gnumeric",
  "file": "fname.sheet"
  }
- **dbi:gnumeric:fname.sheet** (Force Gnumeric interpretation)

MDB: Access database format (via Mdbtools, READ-ONLY)

- **.mdb**: Access database file
- **.json**: {
  "type": "access",
  "file": "fname.db"
  }
- **dbi:access:fname.db** (Force Access interpretation)

MYSQL: database connector

- **.json**: {
  "type": "mysql",
  "database": "db_name",
  "host": "localhost",
  "port": "1111",
  "username": "root",
  "password": "****"
  }
- **dbi:mysql:database_name** (Use default port, username, etc)
- **dbi:mysql:database_name:username=USER:password=PASS**
- **dbi:mysql:database_name:host=HOST:port=PORT**

13.8 Version

ssformat version 0.5.8

14 ssrediff

Reformat a tabular diff file. Converting to hilite/review formats will require supplying the original spreadsheet/database. Not every diff format supported as output by ssdiff can be read as input by ssrediff.

14.1 Usage

- ssrediff [options] PATCHFILE
- ssrediff [options] DATAFILE PATCHFILE

14.2 Index

14.3 Option summary

- --default-table=TABLE
- --format=FORMAT
- --help
- --omit-format-name
- --omit-sheet-name
- --output=OUTPUTFILE
- --patch-formats

14.4 Option details

--default-table=TABLE

name to use when a table name is needed and not supplied

--format=FORMAT

set difference format for output

--help

show how to use this program

--omit-format-name

omit any version-dependent header from diff

--omit-sheet-name

omit any sheet/table name from diff

--output=OUTPUTFILE

direct output to this file (default is standard output)

--patch-formats

list supported patch formats

14.5 Examples

You can generate test file(s) for the examples that follow:

```
ssrediff --test-file numbers.csv
ssrediff --test-file numbers_buggy.csv
```

14.5.1 Example 1

```
ssrediff --format sql numbers_patch.tdiff
```

Convert tdiff format file to SQL

14.5.2 Example 2

```
ssrediff --format csv numbers_patch.tdiff
```

Convert tdiff format file to a CSV-readable diff format

14.5.3 Example 3

```
ssrediff --format hilite --output review.csv numbers_buggy.csv numbers_patch.tdiff
```

Generate tabular form of diff for eyeballing. If ssrediff is compiled with gnumeric support, and output format is ∗.xls, color highlighting is added.

14.6 Patch formats

- **tdiff**: *[default]* vaguely similar to a standard unix diff
- **csv**: csv-compatible diff format
- **hilite**: colorful spreadsheet format
- **index**: tabular output showing relationship between rows and columns
- **raw**: verbose diff format for debugging
- **review**: spreadsheet diff format suitable for quickly accepting or rejecting changes
- **sql**: SQL format (data diffs only)

14.7 Database/spreadsheet file formats

CSV: plain-text delimiter-separated family of formats

- **.csv**: Comma-separated values
- **.tsv**: Tab-separated values
- **.ssv**: Semicolon-separated values

- **.json**: {
  "type": "csv",
  "file": "fname.dsv",
  "delimiter": "|"
  }

SQLITE: file-based database

- **.sqlite**: Sqlite database file
- **.json**: {
  "type": "sqlite",
  "file": "fname.db"
  }
- **dbi:sqlite:fname.db** (Force sqlite interpretation)

SQLITEXT: sqlite-format sql dump

- **.sqlitext**: SQL dump of Sqlite database
- **.json**: {
  "type": "sqlitext",
  "file": "fname.sql"
  }
- **dbi:sqlitext:fname.sql** (Force sqlitext interpretation)

JSONBOOK: Spreadsheet formats in json

- **.jsonbook**: Json spreadsheet

GNUMERIC: Spreadsheet formats (via gnumeric)

- **.xls**: Excel spreadsheet
- **.xlsx**: Excel spreadsheet
- **.gnumeric**: Gnumeric spreadsheet
- **.json**: {
  "type": "gnumeric",
  "file": "fname.sheet"
  }

- **dbi:gnumeric:fname.sheet** (Force Gnumeric interpretation)

MDB: Access database format (via Mdbtools, READ-ONLY)

- **.mdb**: Access database file
- **.json**: {
  "type": "access",
  "file": "fname.db"
  }
- **dbi:access:fname.db** (Force Access interpretation)

MYSQL: database connector

- **.json**: {
  "type": "mysql",
  "database": "db_name",
  "host": "localhost",
  "port": "1111",
  "username": "root",
  "password": "****"
  }
- **dbi:mysql:database_name** (Use default port, username, etc)
- **dbi:mysql:database_name:username=USER:password=PASS**
- **dbi:mysql:database_name:host=HOST:port=PORT**

14.8 Version

ssrediff version 0.5.8

15 ssfossil

This is a thin wrapper around the `fossil` version control system. It modifies the **merge** operation within fossil to use the algorithm behind the ssmerge command.

For usage information, see the `fossil reference`.

The ssfossil client can interoperate with fossil used as a server, or vice versa.

16 coopy

Manage a repository of spreadsheets and databases. Usually run without options, for a graphical interface.

16.1 Usage

- coopy [options]
- coopy [options] DIRECTORY

16.2 Index

16.3 Option summary

- --add=FILE
- --export=FILE
- --gui
- --help
- --key=KEY
- --message=MESSAGE
- --pull
- --push
- --silent

16.4 Option details

--add=FILE

attach the given spreadsheet/database to the repository

--export=FILE

export the given spreadsheet/database from the repository

--gui

force GUI to be shown

--help

show how to use this program

--key=KEY

use specified key when adding or exporting a spreadsheet/database

--message=MESSAGE

use the specified message as a log entry

--pull

pull in data from remote repository to local clone

--push

push out data to remote repository from local clone

--silent

keep output to a minimum

16.5 Examples

16.5.1 Example 1

```
coopy
```

run the coopy GUI from the current directory. All the actions in these examples can be achieved from the GUI.

16.5.2 Example 2

```
coopy --key=people --add=people.xls
```

add people.xls to the repository, with key ’people’.

16.5.3 Example 3

```
coopy --key=orgs --export=organizations.sqlite
```

export organizations.sqlite from the repository, with key 'orgs'.

16.6 Version

coopy version 0.5.8

17 Diff format

- patch_format_tdiff - this is the main supported format. It has the aesthetics of classic diffs.
- patch_format_csv - this is another supported format. It is a CSV readable file.
- patch_format_csv_v_0_2 - an older format, deprecated but still supported.
- Raw format - a verbose format useful for debugging. Cannot be read for patching.

18 Supported table formats

Programs in the COOPY toolbox can work with tables represented in the following formats:

- CSV (or other delimiter-separated formats).
- Sqlite tables.
- MySQL tables (partial support at the time of writing).
- Spreadsheet sheets (.xls, .xlsx, .ods, etc).

CSV support is most complete, followed by Sqlite, followed by Gnumeric, followed by MySQL. Good support for all four formats is targeted for version 1.0 of the toolbox.

18.1 Specifying sources

The COOPY tools expects to work with files as inputs and outputs. For example:

```
ssdiff ver1.sqlite ver2.sqlite
```

To use a database as an input or output, or to add configuration options on how a file should be interpreted, more information needs to be supplied. This can be done in two ways. One is to use a "dbi:*" string. For example:

```
ssdiff ver1.sqlite dbi:mysql:ver2:username=root
```

This would compare the sqlite database in "ver1.sqlite" with the mysql database called "ver2". Alternatively, the extra information can be placed in a "proxy" file with a .json extension, in the JSON format. Here is a file users.json to refer to a table called "users" in a mysql database called "db":

```
{
  "type": "mysql",
  "database": "db",
  "username": "testy",
  "password": "*****",
  "host": "127.0.0.1",
  "port": 3333,
  "table": "users"
}
```

More example dbi strings:

```
dbi:csv:file=hello          # file "hello" is in csv format
dbi:csv::file=hello:there   # file "hello:there" is in csv format
# the use of "::" means that the following key/value pair is the
# last one, and ":" characters should no longer be used as a divider.
```

19 What are all these funny files in a Coopy directory?

There are some special files that fossil uses to store information about a repository. They are:

- _FOSSIL_: this file identifies a directory as a fossil repository.
- manifest: this is a list of all managed files in the repository.
- manifest.uuid: this is a unique identifier. In addition, coopy will place this file in the same directory:
- repository.coopy: this is a database containing information about the repository. In fossil terms, this is the "real" repository, and the rest is just a "view" or "source tree". So as not to confuse people who have not used a version control system before, the coopy program does not emphasize this distinction. The fossil or ssfossil program can be used to create multiple views of the same fossil repository.

20 Related projects

(limited to free and open source projects)

- `diffkit`, diff for tables, targets enterprises.
- `google-diff-match-patch`, for synchronizing plain text.
- `SpreadSheet Compare`, an Excel plugin.