csv2qif convert bank transaction downloads from CSV to QIF

doc generated from the script with csv2qif — bash script, version=2.07

Synopsis

```
csv2qif [options] [ file ...]
```

Options

`-h,--help`	print this help and exit
`-H,--Help`	print full documentation via less and exit
`-V,--version`	print version and exit
`-y,--year=N`	convert data for year N only
`-l,--lower`	convert description field to lower case
`-t,--test`	run in test mode
`-v,--verbose`	print warning for unfound categories

Description

csv2gif converts bank transaction downloads to the QIF format, which can be imported into GnuCash. CSV formats differ from one bank to the other. Currently, the script recognizes ING, RABO and ICS (International Card Services) formats only (on the basis of their field counts, 19 for RABO, 9 for ING, 7 for ICS). All formats may occur in one file.

The ICS data cannot be downloaded from the site, but you can select the data in their display and paste them into a file. This results in a tab- separated file that can be fed to csv2qif.

The QIF formatted data are printed to standard output. Counts of converted lines for each bank account are printed to standard error.

Configuration files

csv2qif looks for the presence of two configuration files and uses the first it encounters:

- `./csv2qif.ods`
- `~/.csv2qif.ods`

These configuration files must be OpenDocument spreadsheets. They can be used to assign records to accounts, based on regular expressions matching descriptions. The files should have three columns:

- The first column contains a sorting character (explained later.)
- the second column contains an account, for example: `Income:Pension`,
- the third column contains one or more bash regular expressions, (see, for example this wiki page). separated by newlines (`ctrl+Enter`) such that, if one of the expressions matches the description, the transaction will be assigned to the `Income:Pension` account.

For example, my `~/.csv2qif.ods` contains the following row, which assigns records to the Expenses:Fuel account:

```
0 Expenses:Fuel    ^bp
                   ^shell
                   tankstation
                   tinq
                   ^tamoil
                   ^total
```

This row says that any record which which starts with `bp␣`, `shell␣`, `tamoil␣` or `total␣`, or contains `tankstation` or `tinq␣`, should be assigned to the `Expenses:Fuel` account. Now if you sometimes also buy clothes at the *Total Body Shop*, which also has the starting word ”total” in its description, your purchases there might go to `Expenses:Fuel`, even if you would have the following row in your configuration

file:

```
0 Expenses:Clothing    total body
                       h&m
```

This is because rows in the configuration file are executed in reverse sorting order. So in thise case, you should change the sorting character for the *Total Body Shop* in, say, 1, so that it will be seen before the `Fuel` row.

Mismatches

If none of the regular expressions in the configuration file matches with the description, an error message is issued on standard error:

```
csv2qif: no cat for prontophot holland b.v. vergoeding foto's 335618587
```

You can then either leave this as it is and, when you import the .qif output into GnuCash, manually set the account. Or you can edit your `csv2qif.ods` file, and enter `prontophot` in the correct list of regular expressions and run **csv2qif** anew.

When, in the startup phase, you have lots of such messages, it may be useful to use the `--test` option. This option suppresses the normal output and filters the error messages through `sort␣-u`, which will hopefully make their interpretation easier.

Author

Wybo Dekker

Copyright

Released under the GNU General Public License