Last update: November 14, 2009

## Welcome to the `umm` web site!

What's `umm`? It's Uwe's Money Mangler — errr, *Manager*! `umm` is a tiny, minimal, command-line accounting program. I was motivated to write it by a couple of things: I had been using Quicken from Intuit for some years, and that had an annoying failure mode every so often: an upgrade of computer hardware or OS would cause Quicken to not run anymore, and some years of my data would thus be trapped in a binary blob and lost. I'd chuck the whole thing in frustration and stop keeping track of this (really rather important) stuff for a year or two, and then I'd start over again. Not ideal!

So... eventually I got tired of this, and I decided to try something different. I had looked at gnucash, but the last time I looked at it I was frankly intimidated... huge, with *lots* of dependencies — although it's open-source, it was not going to be as simple as I wanted. Recently I came across `ledger` and its haskell remix `hledger`, and I looked at those. But there were a couple of minor things I didn't like so much, and there were a couple of features I wanted that they were missing, and I needed a hacking project, so I decided to just take the idea and write my own: `umm`.

## Design Criteria

The design criteria for this little program were fairly simple:

- The data *must* be absolutely portable and transparently accessible, even without `umm` → plain-text file format
  (not entirely coincidentally, this also makes it very easy to save the ledger file in CVS or some other revision control system)
- The program *never* modifies the data — the ledger is *read-only* as far as `umm` is concerned
  (I stole this idea directly from `ledger` and `hledger`; it seems eminently sensible to me)
  You and I, the users, are responsible for editing the ledger, for backing it up, etc etc etc.
- The program should be portable, too — making the program more portable makes the data more portable
- Part of making the program portable is keeping its dependencies to a minimum
  → haskell & standard libraries only, to the extent possible (in practice this means ghc, so far)

A few specific things I wanted the program to do:

- Since I planned on adding several years of data in the past, I didn't want to necessarily be constrained by listing transactions in strict temporal order → order of records is up to the user, `umm` doesn't care. Since there are date stamps on records, `umm` can figure it out
- Sticky notes: lots of times, there are financial things that need to get done at or by a specific date. `umm` can keep track of them and only bug me and thee when it's time to do so
- Unlike `ledger`, I didn't want `umm` to automatically infer the names of new accounts or categories as I enter them:
  I make mistakes, and I don't want those to be perpetuated
- Did I mention that my data must be easily accessible and absolutely portable?

If you prefer a more GUI-oriented program, you should probably look elsewhere... but if you think this might work for you, then read on.

---

### Quick Links

- Grab it here
  - The source code — currently version 0.1.0
  - Executable for iMac G3 running OSX 10.3.9
    at least one report of success on G5 running later version of OSX
  - Executable for Windows XP
- Build & Install
- A Bit of Terminology
- A couple of small examples
- Overview of commands
- Overview of records in the ledger file
- A short tour for people who want to hack on `umm`

### Build `umm` from Source

You'll need a working haskell compiler for this. I use ghc 6.8.3 on an old iMac G3 running OSX 10.3.9, ghc 6.10.1 on x86 running a slightly out-of-date version of Ubuntu Linux, and ghc 6.10.3 on Windows XP

Grab the source, above, unpack it, chdir into the source directory, then run

```
    cabal configure && cabal build
```

or

```
    ghc -O2 --make -o umm UMM*.hs
```

### "Install" `umm`

Whether you built as per above, or you downloaded one of the pre-built versions, just move the `umm` executable to wherever you want, and you're good to go!

### A Bit of Terminology

- ccs — Currency, Commodity, or Security: the stuff you're keeping track of
- account — a bucket which holds varying amounts of different *ccs*
  - you track not just inflows & outflows, but also how much is there at any given time
  - your checking account is a bucket containing just dollars (or euros, or yen, or...)
  - 
  - your brokerage account is a bucket containing dollars and shares of various stocks, mutual funds, etc
  - your mattress is a bucket containing ounces of gold, or pork bellies (don't tell the health department!)
- pseudo-account — a source or sink of *ccs*, such as "interest"
  - you don't necessarily want to track how much interest there is overall,
    but you do want to track where that money came from
  - you can think of this as a category in Quicken
  - if you want to do true double-entry accounting, you can simply not use pseudo-accounts:

set up a true account for each category, and you're set

### A Couple of Small Examples

Here's a complete small ledger file (sample1.dat in the source tarball):

```
ccs US$ "The Almighty Buck"

income cash
income interest
expense cash

account acc1 2009-9-1 "test account 1"
account acc2 2009-9-1 "test account 2"

group all acc1 acc2

ccs FOO "The Foo Companies"
ccs BAR "The Bar Companies"
ccs CLAM "Groupe Mollusque"

xfer 2009-9-1 cash acc1 100
buy 2009-9-1 acc1 10 FOO 15
xfer 2009-9-2 interest acc1 0.25
buy 2009-9-2 acc1 20 FOO 27
xfer 2009-9-3 acc1 cash 5.25

xfer 2009-9-1 cash acc2 100
buy 2009-9-1 acc2 10 BAR 25
xfer 2009-9-2 interest acc2 0.25
buy 2009-9-2 acc2 20 FOO 27
xfer 2009-9-3 acc2 cash 5.25
xfer 2009-9-4 acc1 acc2 25

xfer 2009-9-5 cash acc1 200
buy 2009-9-5 acc1 30 FOO 40
buy 2009-9-5 acc1 30 BAR 40

price 2009-9-5 FOO 1.5

xfer 2009-9-6 cash acc1 20 CLAM

price 2009-10-21 FOO 2.1
price 2009-10-24 FOO 2.4
price 2009-10-25 FOO 2.5
price 2009-10-23 FOO 2.3
price 2009-10-22 FOO 2.2

price 2009-10-25 BAR 3.25
price 2009-10-27 3 BAR 10

split 2009-11-5 CLAM 2 1
price 2009-11-11 CLAM 0.001
```

### Overview of Commands

You run the program like so:

`umm` *ledger-file command options*

where *command options* is one of the following

- `balance` [*account-or-group*] [*date*]
  shows the balance in the specified account or group, or in all accounts if none is specified, as of the specified date.

- `register` *account* [*date*] [*date*]
  shows all transactions involving the specified account up to the specified date, and shows the balance as of that date.
  If two dates are specified, only transactions in the interval between the two dates are printed.

- `reconcile` [*account*] [*date*]
  applies all reconciled transactions up to the specified date, shows relevant unreconciled transactions up to that date,
  and shows the reconciled balance(s) as of that date.

  If an account is specified, then only unreconciled transactions involving that account are shown, otherwise all unreconciled transactions are shown.

- `change` *account-or-pseudo-account* [*date*] [*date*]
  shows the change in the specified account or pseudo-account between the two specified dates,
  or from the beginning of the ledger to the specified date.

- `price` *ccs* [*date*]
  shows the price history of the specified *ccs* up to the specified date.

- `todo` [*date*]
  shows all the unreconciled *todo* items in the ledger up to the specified date.

- `list` [`all` | `accounts` | `ccs` | `expenses` | `income` | `groups`]
  shows summaries of the various types of non-transaction entries in the ledger file.
  If no type is specified, all are shown

- `basis` *ccs* [*date*]
  shows the cost basis for a given currency or commodity or security as of the specified date.

The commands may be shortened to the unique prefixes or anything longer: 'bal', 'bala' etc, 'reg', 'regi' etc, but not 'r'.
If no command or options are specified, the default action is to show the balances of all accounts as of the current date.

*account* is the name of an account, *account-or-group* is the name of an account or an account group,
and *account-or-pseudo-account* is the name of an account or an income or expense pseudo-account.

*ccs* is the name of a currency, commodity, or security.

*date* defaults to the current date if not specified.

## Overview of Records

*under construction* — see the actual program, which does have a description of each record type in its help listing.
Also, the records parser, in `UMMParser.hs`, reads a lot like a BNF description of the grammar, thanks to the Parsec library.

### Overview of Code, for Hackers

*under construction*

---

If you have questions or comments about the stuff here, please contact me at korg@korgwal.com.

Enjoy! -- Uwe Hollerbach

Page last modified at 2009/11/14 21:30 US/Pacific