Ledger: Command-Line Accounting

John Wiegley

April 17, 2004

Contents

Introduction

Ledger is an accounting tool with the moxie to exist. It provides no bells or whistles, and returns the user to the days before user interfaces were even a twinkling in their father's CRT.

What it does do is to offer a double-entry accounting ledger with all the flexibility and muscle of its modern day cousins, without any of the fat. Think of it as the Great Bran Muffin of accounting tools.

To use it, you need to start keeping a ledger. This is the basis of all accounting, and if you haven't started yet, now is the time to learn. The little booklet that comes with your checkbook is a ledger, so we'll describe double-entry accounting in terms of that.

A checkbook ledger records debits (subtractions, or withdrawals) and credits (additions, or deposits) with reference to a single account: the checking account. Where the money comes from, and where it goes to, are described in the payee field, where you write the person or company's name. The ultimate aim of keeping a checkbook ledger is to know how much money is available to spend. That's really the aim of all ledgers.

What computers add is the ability to walk through these transactions, and tell you things about your spending habits; to let you devise budgets and get control over your spending; to squirrel away money into virtual savings account without having to physically move money around; etc. As you keep your ledger, you are recording information about your life and habits, and sometimes that information can start telling you things you aren't aware of. Such is the aim of all good accounting tools.

The next step up from a checkbook ledger, is a ledger that keeps track of all your accounts, not just checking. In such a ledger, you record not only who gets paid—in the case of a debit—but where the money came from. In a checkbook ledger, its assumed that all the money comes from your checking account. But in a general ledger, you write transaction two-lines: The source account and target account. *There must always be a debit from at least one account for every credit made to another account.* This is what is meant by "double-entry" accounting: the ledger must always balance to zero, with an equal number of debits and credits.

For example, let's say you have a checking account and a brokerage account, and you can write checks from both of them. Rather than keep two checkbooks, you decide to use one ledger for both. In this general ledger you need to record a payment to Pacific Bell for your monthly phone bill. The cost is $23.00, let's say, and you want to pay it from your checking account. In the general ledger you need to say where the money came from, in addition to where it's going to. The entry might look like this:

```
9/29  BAL  Pacific Bell                $200.00    $200.00
           Equity:Opening Balances    $-200.00
9/29  BAL  Checking                    $100.00    $100.00
           Equity:Opening Balances    $-100.00
9/29  100  Pacific Bell                 $23.00    $223.00
           Checking                    $-23.00     $77.00
```

The first line shows a credit (or payment) to Pacific Bell for $23.00. Because there is no "balance" in a general ledger—it's always zero—we write in the total balance of all payments to "Pacific Bell", which now is $223.00 (previously the balance was $200.00). This is done by looking at the last entry for "Pacific Bell" in the ledger, adding $23.00 to that amount, and writing the total in the balance column. And the money came from "Checking"—a debit (or withdrawal) of $23.00—which leaves the ending balance in "Checking" at $77.00. This is a very manual procedure; but that's where computers come in...

The transaction must balance to $0: $23 went to Pacific Bell, $23 came from Checking. There is nothing left over to be accounted for, since the money has simply moved from one account to another. This is the basis of double-entry accounting: that money never pops in or out of existence; it is always a transaction from one account to another.

Keeping a general ledger is the same as keeping two separate ledgers: One for Pacific Bell and one for Checking. In that case, each time a credit is written into one, you write a corresponding debit into the other. This makes it easier to write in a "running balance", since you don't have to look back at the last time the account was referenced—but it also means having a lot of ledger books, if you deal with multiple accounts.

Enter the beauty of computerized accounting. The purpose of the Ledger program is to make general ledger accounting simple, by keeping track of the balances for you. Your only job is to enter the credit/debit transactions. If a transaction does not balance, Ledger will display an error and indicate which transaction is wrong.[1]

In summary, there are two aspects of Ledger use: Updating the ledger data file, and using the Ledger tool to view the summarized result of your entries.

[1]In some special cases, it will automatically balance the entry for you.

Building the program

Ledger is written in ANSI C++, and should compile on any platform. It depends only on the GNU multiprecision integer library (libgmp), and the Perl regular expression library (libpcre). It was developed using GNU make and gcc 3.3.

To build and install once you have these libraries on your system, enter these commands:

```
make
cp ledger /usr/local/bin
```

Keeping a ledger

The most important part of accounting is keeping a good ledger. If you have a good ledger, tools can be written to work whatever mathematically tricks you need to better understand your spending patterns. Without a good ledger, no tool, however smart, can help you.

The Ledger program aims at making ledger entry as simple as possible. Since it is a command-line tool, it does not provide a user interface for keeping a ledger. If you like, you may use Gnucash to maintain your ledger, in which case the Ledger program will read Gnucash's data files directly. In that case, read the Gnucash manual now, and skip to the next chapter.

If you are not using Gnucash, but a text editor to maintain your ledger, read on. Ledger has been designed to make data entry as simple as possible, by keeping the ledger format easy, and also by automagically determining as much information as possible based on the nature of your entries.

For example, you do not need to tell Ledger about the accounts you use. Any time Ledger sees a debit or a credit to an account it knows nothing about, it will create it. If you use a commodity that is new to Ledger, it will create that commodity, and determine its display characteristics (placement of the symbol before or after the amount, display precision, etc) based on how you used the commodity in the transaction.

Here is the Pacific Bell example from above, given as a Ledger transaction:

```
9/29  (100)  Pacific Bell
    Expenses:Utilities:Telephone               $23.00
    Assets:Checking                           $-23.00
```

As you can see, it is very similar to what would be written on paper, minus the computed balance totals, and adding in account names that work better with Ledger's scheme of things. In fact, since Ledger is smart about many things, you don't need to specify the balanced amount, if it is the same as the first line:

```
9/29  (100)  Pacific Bell
    Expenses:Utilities:Telephone                 $23.00
    Assets:Checking
```

For this entry, Ledger will figure out that $-23.00 must come from "Assets:Checking" in order to balance the entry.

Credits and Debits

Credit and debit are simple enough terms in themselves, but the usages of the modern world have made them very hard to puzzle out.

Basically, a credit means you add something to an account, and a debit means you take away. A debit card is correctly name: From your point of view, it debits your checking account every time you use it.

The credit card is strangely named, because you have to look at it from the merchant's point of view: Every time you use it, it credit's *his* account right away. This was a giant leap from the days of cash and checks, when the only other way to supply immediate credit was by a wire transfer. But a credit card does not credit you anything at all. In fact, from your point of view, it should be called a liability card, since it increases your liability to the issuing bank every time you use it.

In Ledger, credits and debits are given as they are, which means that sometimes you will see a minus sign where you don't expect one. For example, when you get paid, in order to credit your bank account, you need to debit an income account:

```
9/29  My Employer
    Assets:Checking                               $500.00
    Income:Salary                                $-500.00
```

But wait, you say, why is the Income a negative figure? And when you look at the balance totals for your ledger, you will certainly be surprised to see Expenses as a positive figure, and Income as a negative figure. Isn't that the opposite of how it should look?

It may take getting used to, but to properly use a general ledger you will need to think in terms of correct debits and credits. Rather than Ledger "fixing" the minus signs, let's understand why they are there.

When you earn money, the money has to come from somewhere. Let's call that somewhere "society". In order for society to give you an income, you must take money away from society (debit) in order to put it into your bank (credit). When you then spend that money, it leaves your bank account (debit) and goes back to society (credit). This is why Income will appear negative—it reflects the money you have drawn from society—and why Expenses will be positive—it is the amount you've given back. These credits and debits will always cancel each other out in the end, because you don't have the ability to create new money: It must always come from somewhere, and in the end must always leave. This is the beginning of economy, after which the explanation gets terribly difficult.

Based on that explanation, here's another way to look at your balance report: Every negative figure means that that account or person or place has less money now than when you started your ledger; and every positive figure means that that account or person or place has more money now that when you started your ledger. Make sense?

Also, credit cards will have a negative value, because you are spending *from* them (debit) in order pay someone else (credit). They are called credit cards because you are able to instantly credit that other person, by simply waving a card.

Commodities and Currencies

Ledger makes no assumptions about the commodities you use; it only requires that you specify a commodity. The commodity may be any non-numeric string that does not contain a period, comma, forward slash or at-sign. It may appear before or after the amount, although it is assumed that symbols appearing before the amount refer to currencies, while non-joined symbols appearing after the amount refer to commodities. Here are some valid currency and commodity specifiers:

```
$20.00         ; currency: twenty US dollars
USD 20         ; currency: the same
40 AAPL        ; commodity: 40 shares of Apple stock
MD 60          ; currency: 60 Deutsch Mark
£50            ; currency: 50 British pounds
```

```
50e             ; currency: 50 Euros (use symbol)
```

Ledger will examine the first use of any commodity to determine how that commodity should be printed on reports. It pays attention to whether the name of commodity was separated from the amount, whether it came before or after, the precision used in specifying the amount, whether thousand marks were used, etc. This is done so that printing the commodity looks the same as the way you use it.

An account may contain multiple commodities, in which case it will have separate totals for each. For example, if your brokerage account contains both cash, gold, and several stock quantities, the balance might look like:

```
  $200.00
100.00 AU
  AAPL 40
 BORL 100
 FEQTX 50  Assets:Brokerage
```

This balance report shows how much of each commodity is in your brokerage account.

Sometimes, you will want to know the current street value of your balance, and not the commodity totals. For this to happen, you must specify what the current price is for each commodity. The price can be in any commodity, in which case the balance will be computed in terms of that commodity. The usual way to specify prices is with a file of price settings, which might look like this:

```
AU=$357.00
AAPL=$37
BORL=$19
FEQTX=$32
```

Specify the prices file using the `-p` option:

```
ledger -p prices.db balance brokerage
```

Now the balance for your brokerage account will be given in US dollars, since the prices database has specified conversion factors from each commodity into dollars:

```
    $40880.00  Assets:Brokerage
```

You can convert from any commodity to any other commodity. Let's say you had $5000 in your checking account, and for whatever reason you wanted to know many ounces of gold that would buy. If gold is currently $357 per ounce, then each dollar is worth 1/357 AU:

```
    ledger -p "$=0.00280112 AU" balance checking

    14.01 AU  Assets:Checking
```

$5000 would buy 14 ounces of gold, which becomes the new display commodity since a conversion factor was provided.

Commodities conversions can also be chained, up to a depth of 10. Here is a sample prices database that uses chaining:

```
    AAPL=$15
    $=0.00280112 AU
    AU=300 Euro
    Euro=MD 0.75
```

This is a roundabout way of reporting AAPL shares in their Deutsch Mark equivalent.

Accounts and Inventories

Since Ledger's accounts and commodity system is so flexible, you can have accounts that don't really exist, and use commodities that no one else recognizes. For example, let's say you are buying and selling various items in EverQuest, and want to keep track of them using a ledger. Just add items of whatever quantity you wish into your EverQuest account:

```
    9/29  Get some stuff at the Inn
        Places:Black's Tavern                   -3 Apples
        Places:Black's Tavern                   -5 Steaks
        EverQuest:Inventory
```

Now your EverQuest:Inventory has 3 apples and 5 steaks in it. The amounts are negative, because you are taking *from* Black's Tavern in order to credit your Inventory account. Note that you don't have to use "Places:Black's Tavern" as the source account. You could use "EverQuest:System" to represent the fact that you acquired them online. The only purpose for choosing one kind of source account over another is for generate more informative reports later on. The more you know, the better analysis you can perform.

If you later sell some of these items to another player, the entry would look like:

```
10/2  Strum Brightblade
    EverQuest:Inventory                    -2 Steaks
    EverQuest:Inventory                    15 Gold
```

Now you've turned 2 steaks into 15 gold, courtesy of your customer, Strum Brightblade.

Understanding Equity

The most confusing entry in any ledger will be your equity account— because starting balances can't come out of nowhere.

When you first start your ledger, you will likely already have money in some of your accounts. Let's say there's $100 in your checking account; then add an entry to your ledger to reflect this amount. Where will money come from? The answer: your equity.

```
10/2  Opening Balance
    Assets:Checking                        $100.00
    Equity:Opening Balances
```

But what is equity? You may have heard of equity when people talked about house mortgages, as "the part of the house that you own". Basically, equity is like the value of something. If you own a car worth $5000, then you have $5000 in equity in that car. In order to turn that car (a commodity) into a cash flow, or a credit to your bank account, you will have to debit the equity by selling it.

When you start a ledger, you are probably already worth something. Your net worth is your current equity. By transferring the money in the ledger from your equity to your bank accounts, you are crediting the ledger account based on your prior equity value. That is why, when you look at the balance report, you will see a large negative number for Equity that never changes: Because that is what you were worth (what you debited from yourself in order to start the ledger) before the money started moving around. If the total positive value of your assets is greater than the absolute value of your starting equity, it means you are making money.

Clear as mud? Keep thinking about it. Until you figure it out, put "– - Equity" at the end of your balance command, to remove the confusing figure from the totals.

Dealing with cash

Something that stops many people from keeping a ledger at all is the insanity of tracking cash expenses. They rarely generate a receipt, and there are often a lot of small transactions, rather than a few large ones, as with checks.

The answer is: don't bother. Move your spending to a debit card, but in general ignore cash. Once you withdraw it from the ATM, mark it as already spent to an "Expenses:Cash" category:

```
2004/03/15 ATM
    Expenses:Cash                          $100.00
    Assets:Checking
```

If at some point you make a large cash expense that you want to track, just "move" the amount of the expense from "Expenses:Cash" into the target account:

```
2004/03/20 Somebody
    Expenses:Food                           $65.00
    Expenses:Cash
```

This way, you can still track large cash expenses, while ignoring all of the smaller ones.

Virtual transactions

A virtual transaction is when you, in your mind, see money as moving to a certain place, when in reality that money has not moved at all. There are several scenarios in which this type of tracking comes in handy, and each of them will be discussed in detail.

To enter a virtual transaction, surround the account name in parentheses. This form of usage does not need to balance. However, if you want to ensure the virtual transaction balances with other virtual transactions in the same entry, use square brackets. For example:

```
10/2  Paycheck
    Assets:Checking                        $1000.00
    Income:Salary                         $-1000.00
    (Debt:Alimony)                          $200.00
```

In this example, after receiving a paycheck an alimony debt is increased—even though no money has moved around yet.

```
10/2  Paycheck
    Assets:Checking                        $1000.00
    Income:Salary                         $-1000.00
    [Savings:Trip]                          $200.00
    [Assets:Checking]                      $-200.00
```

In this example, $200 has been deducted from checking toward savings for a trip. It will appear as though the money has been moved from the account into "Savings:Trip", although no money has actually moved anywhere.

When balances are displayed, virtual transactions will be factored in. To view balances without any virtual balances factored in, using the "-R" flag, for "Reality".

Write about: Saving for a Special Occasion; Keeping a Budget; Tracking Allocated Funds.

Automated transactions

As a Bahá'í, I need to compute Huqúqu'lláh whenever I acquire assets. The exact details of this are a bit complex, so if you have further interest, please consult the Web.

For any fellow Bahá'ís out there who want to track Huqúqu'lláh, the Ledger tool makes this extremely easy. Just set up the following automated transaction at the top of your ledger file:

```
; These entries will compute Huqúqu'lláh based on the
; contents of the ledger.
```

```
= ^Income:
= ^Expenses:Rent$
= ^Expenses:Furnishings
= ^Expenses:Business
= ^Expenses:Taxes
= ^Expenses:Insurance
  (Liabilities:Huqúqu'lláh)                0.19
```

This automated transaction works by looking at each transaction appearing afterward in the ledger file. If any match the account regexps, occurring after the equal signs above, 19that transaction is applied to the "Liabilities:Huqúqu'lláh" account. So if $1000 is earned through Income:Salary, which is seen as a debit from Income, a debit of $190 is applied to "Liabilities:Huqúqu'lláh"; if $1000 is spent on Rent—seen as a credit to the Expense account—a credit of $190 is applied to Huqúqu'lláh. The ultimate balance of Huqúqu'lláh reflects how much must be paid to that account in order to balance it to zero.

When you're ready to pay, just write a check directly to the account "Liabilities:Huqúqu'lláh":

```
2003/01/01 (101) Baha'i Huqúqu'lláh Trust
    Liabilities:Huqúqu'lláh         $1,000.00
    Assets:Checking
```

That's it. To see how much Huqúq is currently owed based on your ledger entries, use:

```
ledger balance Liabilities:Huqúq
```

Running Ledger

Now that you have an orderly and well-organized general ledger, it's time to start generating some orderly and well-organized reports. This is where the Ledger tool comes in. With it, you can balance your checkbook, see where your money is going, tell whether you've made a profit this year, and even compute the present day value of your retirement accounts. And all with the simplest of interfaces: the command-line.

The most often used command will be the "balance" command:

```
export LEDGER=/home/johnw/doc/ledger.dat
ledger balance
```

Here I've set my Ledger environment variable to point to where my ledger file is hiding. Thereafter, I needn't specify it again.

The balance command prints out the summarized balances of all my top-level accounts, excluding sub-accounts. In order to see the balances for a specific account, just specify a regular expression after the balance command:

```
ledger balance expenses:food
```

This will show all the money that's been spent on food, since the beginning of the ledger. For food spending just this month (September), use:

```
ledger -d sep balance expenses:food
```

Or maybe I want to see all of my assets, in which case the -s (show sub-accounts) option comes in handy:

```
ledger balance -s
```

To exclude a particular account, use a regular expression with a leading minus sign. The following will show all expenses, but without food spending:

```
ledger balance expenses -food
```

If you want to show all accounts but for one account, remember to use "–" to separate the exclusion pattern from the options list:

```
ledger balance -- -equity
```

Command summary

balance

The "balance" command reports the current balance of any account. This command accepts a list of optional regexps, which will confine the balance report to only matching accounts. By default, the balances for all accounts will be printed. If an account contains multiple types of commodities, each commodity's total is separately reported.

register

The "register" command displays all the transactions occurring in a single account, line by line. The account regexp must be specified as the only argument to this command. If any regexps occur after the required account name, the register will contain only those transactions that match. Very useful for hunting down a particular transaction.

The output from "register" is very close to what a typical checkbook, or single account ledger, would look like. It also shows a running balance. The final running balance of any register should always be the same as the current balance of that account.

print

The "print" command prints out ledger entries just as they appear in the original ledger. They will be properly formatted, and output in the most economic form possible. The "print" command also takes a list of optional regexps, which will cause only those transactions which match in some way to be printed.

The "print" command is a handy way to clean up a ledger file whose formatting has gotten out of hand.

equity

Equity transactions are used to establish the starting value of an account. You might think of equity as the "ether" from which initial balances appear.

The "equity" command makes it easy to archiving past years, and then remove them without changing any current balances. For example, if it's now 2004 and we want to archive all of 2003's transactions to another file, write:

```
export LEDGER=ledger.dat
ledger -e 2004/1/1 print  > ledger-2003.dat
ledger -e 2004/1/1 equity > /tmp/balances
ledger -b 2004/1/1 print  > /tmp/current
cat /tmp/balances /tmp/current > ledger.dat
rm /tmp/balances /tmp/current
```

After these commands, **ledger-2003.dat** will contain all the transactions up to year 2004, with **ledger.dat** containing only those since 2004. However, the balances reported from **ledger.dat** will still be the same.

Sometimes you will not want to carry forward certain balances, such as those for Expense and Income. To do this, change the second command above to:

```
ledger -e 2004/1/1 equity -^Income -^Expenses > /tmp/balances
```

entry

The three most laborious tasks of keeping a ledger are: adding new entries, reconciling accounts, and generating reports. To address the first of these, there is a sub-command to ledger called "entry". It works on the principle that 80earlier transactions. Here's how it works:

Let's say you have an old transaction of the following form:

```
2004/03/15 * Viva Italiano
    Expenses:Food                       $12.45
    Expenses:Tips                        $2.55
    Liabilities:MasterCard             $-15.00
```

Now it's 2004/4/9, and you've just eating at Viva Italiano again. The exact amounts are different, but the overall form is the same. With the "entry" command you can type:

```
ledger entry 2004/4/9 viva food 11.00 tips 2.50
```

This will produce the following output:

```
2004/04/09 Viva Italiano
    Expenses:Food                       $11.00
    Expenses:Tips                        $2.50
    Liabilities:MasterCard             $-13.50
```

This works by finding a transaction that matches the regexp "viva", and then assuming that any accounts or amounts you specify will be the same as that earlier transaction. If Ledger does not succeed in generating a new entry for you, it will print an error and set the exit code to 1.

There is a shell script in the distribution called "entry", which simplifies the task of adding a new entry to your ledger, and then launches `vi` to let you confirm that the entry looks appropriate.

Option summary

- Only consider entries occuring on or after the given date.
- Only consider entries occuring before the given date. The date is not inclusive, so any entries occurring on that date will not be used.
- Only consider entries occurring on or before the current date.
- Only consider entries whose cleared flag has been set. The default is to consider both.
- Only consider entries fitting the given date mask. DATE in this case may be the name of a month, or a year, or a year and month, such as "2004/05". It's a shorthand for having to specify -b and -e together.
- Read ledger entries from FILE. This takes precedence over the environment variable LEDGER.
- Print full account names in all cases, such as "Assets:Checking" instead of just "Checking". Only used current by the "balance" command.
- Print out quick help on the various options and commands.
- Read in the list of patterns to include/exclude from FILE. Ordinarily, these are specified as arguments after the command.
- When used with the "register" command, causes only monthly subtotals to appear. This can be useful for looking at spending patterns. TODO: Accept an argument which specifies the period to use.
- Modifies the output generated by -M to be friendly to programs like Gnuplot. It strips away the commodity label, and outputs only two columns: the date and the amount.
- Do not show subtotals in the balance report, or split transactions in the register report.
- If a string, such as "COMM=$1.20", the commodity COMM will be reported only in terms of its translated dollar value. This can be used to perform arbitrary value substitutions. For example, to report the value of your dollars in terms of the ounces of gold they would buy, use: -p "$=0.00280112 AU" (or whatever the current exchange rate is).

- Download current prices for all commodities by calling the script “getquote”. There is a “getquote” script included with ledger, although any similar program could be used. It must take a single argument, the name of the commodity, and must report the value as a single amount, such as would appear in a ledger file. If the commodity has no price, nothing should be output and the exit code should be set to a non-zero value.

- Ignore all virtual transactions, and report only the real balance for each account.

- If an account has children, show them in the balance report.

- Sort the ledger after reading it. This may affect “register” and “print” output.

- Show only uncleared transactions. The default is to consider both.

- Display the version of ledger being used.

Using Emacs to Keep Your Ledger

In the Ledger tarball is an Emacs module, `ledger.el`. This module makes the process of keeping a text ledger much easier for Emacs users. I recommend putting this at the top of your ledger file:

```
; -*-ledger-*-
```

And this in your `.emacs` file, after copying `ledger.el` to your site-lisp directory:

```
(load "ledger")
```

Now when you edit your ledger file, it will be in `ledger-mode`. `ledger-mode` adds the following commands:

C-c C-a For quickly adding new entries based on the form of older ones (see previous section).

C-c C-c Toggles the “cleared” flag of the transaction under point.

C-c C-r Reconciles an account by displaying the transactions in another buffer, where simply hitting the spacebar will toggle the cleared flag of the transaction in the ledger. It also displays the current cleared balance for the account in the modeline.

Using GnuCash to Keep Your Ledger

The Ledger tool is fast and simple, but it gives you no special method of actually editing the ledger. It assumes you know how to use a text editor, and like doing so. Perhaps an Emacs mode will appear someday soon to make editing Ledger's data files much easier.

Until then, you are free to use GnuCash to maintain your ledger, and the Ledger program for querying and reporting on the contents of that ledger. It takes a little longer to parse the XML data format that GnuCash uses, but the end result is identical.

Then again, why would anyone use a Gnome-centric, 35 megabyte behemoth to edit their data, and a 65 kilobyte binary to query it. . .