Dynamic Ledger: a tutorial

December 8, 2013

Will Landau

Department of Statistics
Iowa State University

Contents

# 1 Introduction and basic usage

Dynamic Ledger is a standalone C program for managing personal finances. Unlike most other accounting software, it takes into account transaction delays and multiple credit accounts when it computes summaries. It also allows the user to easily condense the ledger periodically so that the file remains lightweight over time.

According to the program's conceptual model, transactions flow through credit accounts and eventually arrive in partitions of bank accounts (i.e., Food, Bills, Medical, etc.).
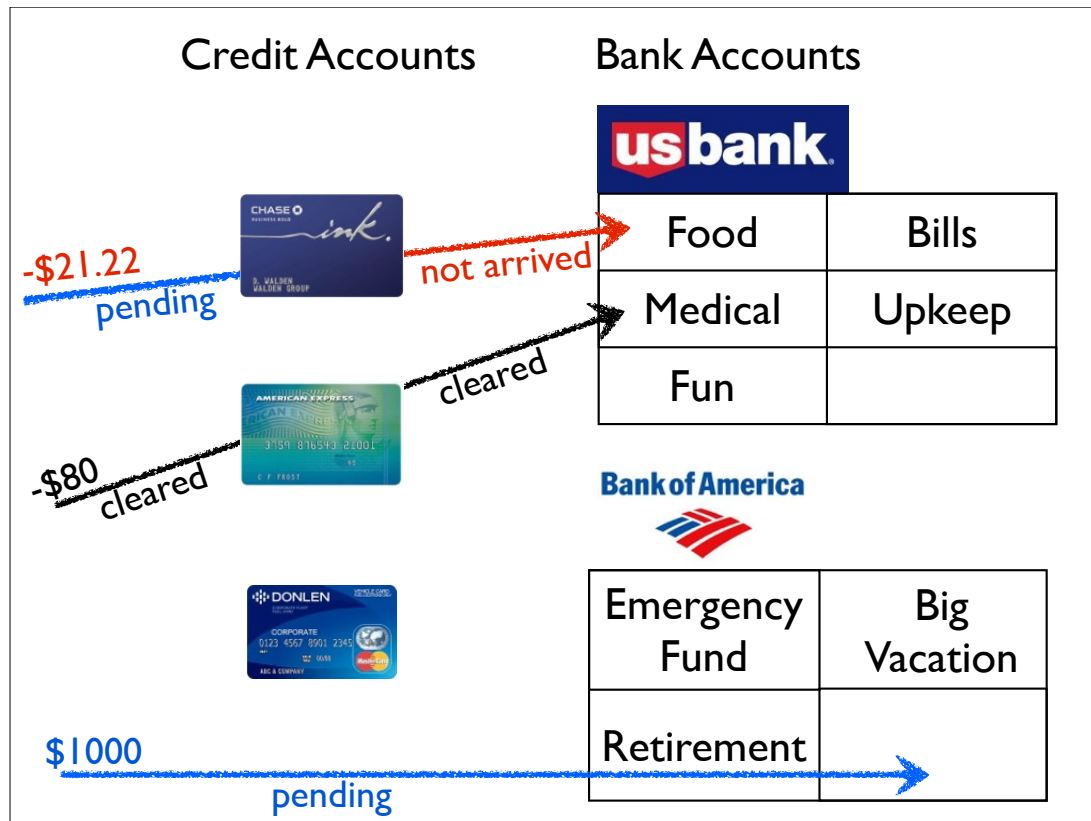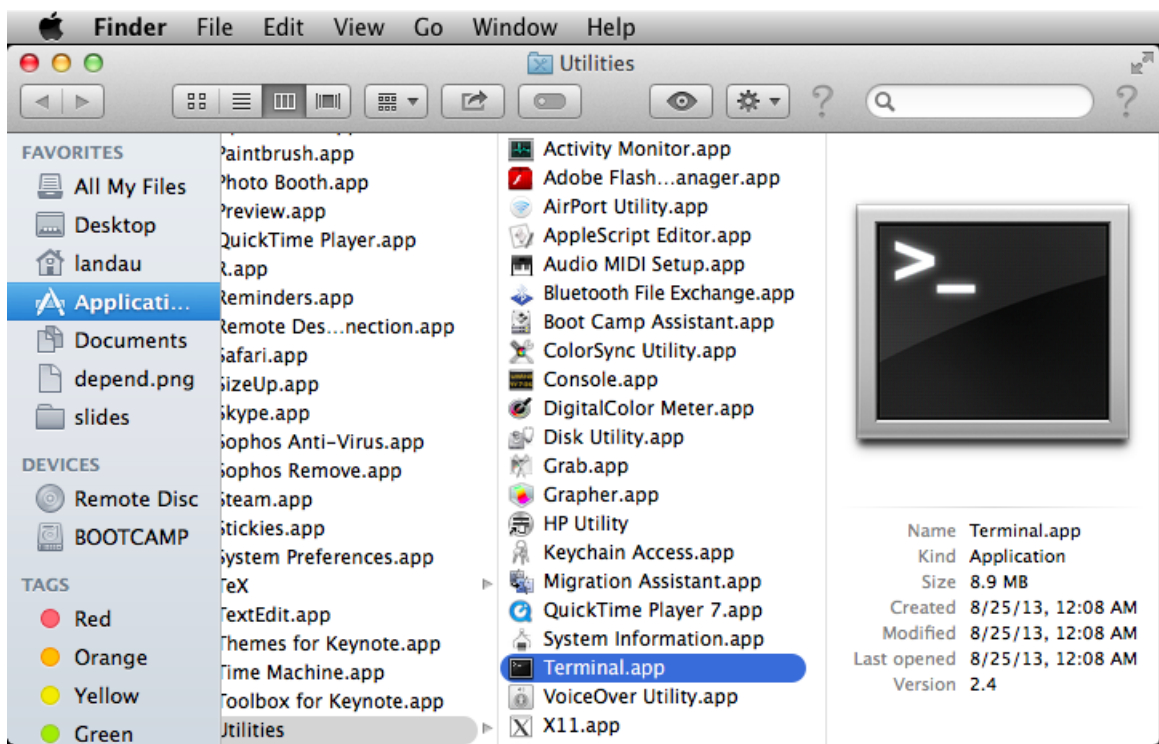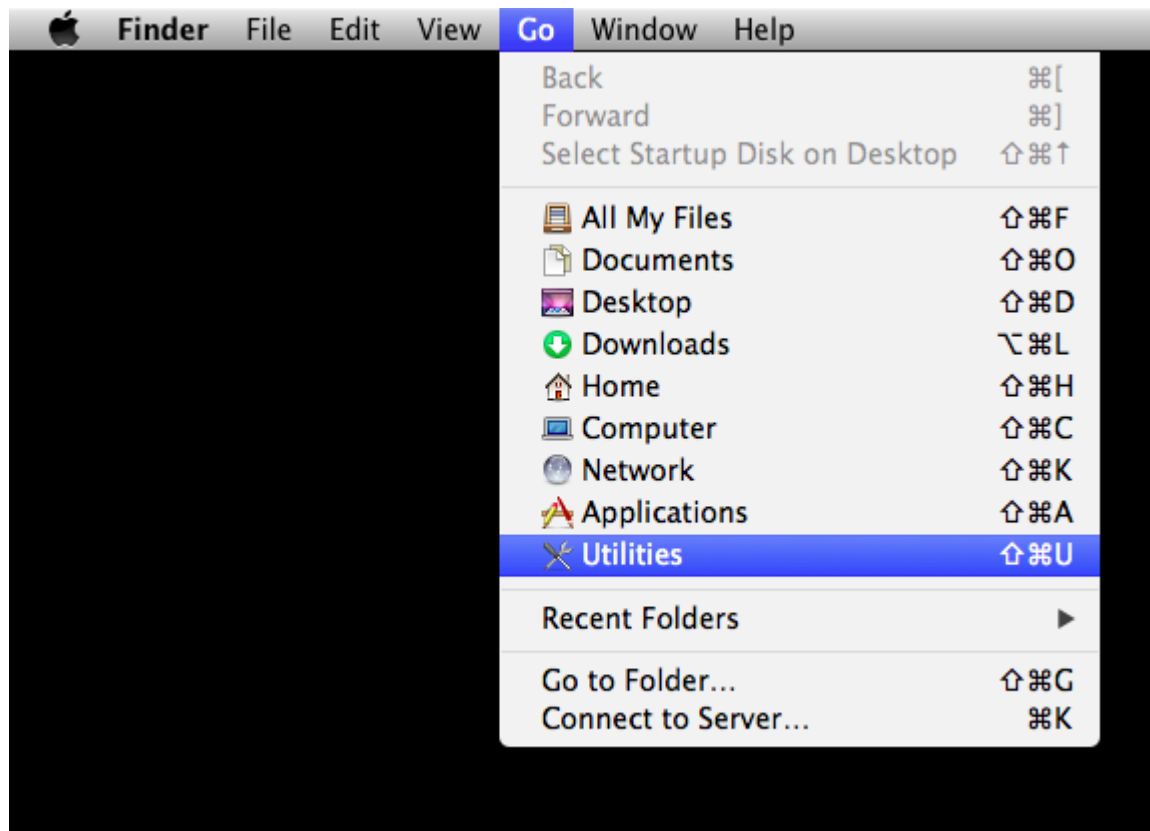

The amount, delay status, credit account, and bank account of each transaction are recorded in a tab-delimited file. For example,

example_ledger1.txt					
amount	status	credit	bank	partition	description
−21.22	cp	chase ink	us bank	food	groceries 12/7/13
−80		am ex	us bank	medical	coinsurance
1000		bank of am			misc savings

Note that "cp" in the status column indicates a transaction that is pending with respect to its credit account and still completely absent from its final destination bank account. For more information on transaction status codes, please see Section 4.

To output a summary of `example_ledger1.txt` on a Mac, I first locate and open Terminal.

Then, I compile and run the program in a Terminal window.

The output gives the true balance of each account, along with other summaries that reflect transaction delays. Note that the program ignores accounts and partitions with balances of zero.

For the next major feature of the program, consider a longer ledger file.

amount	status	credit	bank	partition	description
example_ledger2.txt					
amount	status	credit	bank	partition	description
−30.14	cn	card1	checking	food	groceries 12/3/13
−15.36	cn	card2	checking	food	produce 12/2/13
−10.41	cp	card1	checking	food	dinner 11/21/13
−7.31	c	card2	checking	fun	movie 11/18/13
−500	p		checking	bills	December rent
−30	l		checking	bills	November electric
−50	l		checking	bills	November heat
−45			checking	bills	The Economist
800	p		checking	rent	November paycheck

400	p		checking	food	November paycheck
500	p		checking	upkeep	November paycheck
50	p		checking	fun	November paycheck
50			checking	fun	won $50 in poker
200			savings	bike	gift money
200			savings	bike	October paycheck
100			savings		general savings
200					cash

The summary output for this file looks like

I may wish make a shorter ledger by condensing all the "unlocked" (status ≠ "l") transactions that have cleared. In that case, I can simply run the program, but specify an output file.

The output, `condensed_ledger.txt`, contains only the locked transactions, delayed transactions, and bank partition totals.

amount	status	credit	bank	partition	description
−30.14	cn	card1	checking	food	groceries 12/3/13
−15.36	cn	card2	checking	food	produce 12/2/13
−10.41	cp	card1	checking	food	dinner 11/21/13
−7.31	c	card2	checking	fun	movie 11/18/13
−500.00	p		checking	bills	December rent
−30.00	l		checking	bills	November electric
−50.00	l		checking	bills	November heat
800.00	p		checking	rent	November paycheck
400.00	p		checking	food	November paycheck
500.00	p		checking	upkeep	November paycheck
50.00	p		checking	fun	November paycheck
−45.00			checking	bills	condensed
50.00			checking	fun	condensed
400.00			savings	bike	condensed
100.00			savings		condensed
200.00					condensed

The summary output of `condensed_ledger.txt` agrees with that of `example_ledger2.txt`.

2 Installation requirements

- A command line interface program like Terminal. Windows users will need to download Cygwin or MinGW. (Command Prompt is not sufficient.)

- The GNU Compiler Collection. Mac users should obtain it by installing Xcode command line tools. Windows users should be able to obtain it through Cygwin or MinGW.

3 Data format

Roughly speaking, each row in a ledger file represents a transaction, and each column represents a feature of that transaction. Here are more details about the columns.

amount   Amount of money transferred in the transaction. All entries must be legal numbers. Negative numbers represent payments, and positive numbers represent deposits.

status   Transaction status codes, explained later, reflect the delay status of each transaction. A blank status indicates that the transaction has cleared. You can also use the status column to label any cleared transactions that you want to remain untouched in a condensed ledger. Warning: all unrecognized status codes will be ignored and those transactions will be treated as cleared.

credit   Names of the credit accounts.

bank   Names of the bank accounts.

partition   Names of the bank account partitions. It is useful to divide your accounts into different partitions: for example, food, medical, rent, etc.

description   A short memo describing the transaction, possibly including the merchant and the date. This field is the most flexible and it will not be processed by the program.

4   Transaction status codes

Transaction status codes are listed in the "status" column of a ledger file, and they reflect the delay status of transactions. They are defined in `include/user_settings.h`.

```
#define  CREDIT_NOT_THERE_YET  "cn"   /*** HASN'T REACHED CREDIT COMPANY YET ***/
#define  CREDIT_PENDING        "cp"   /*** PENDING CHARGE IN CREDIT ACCOUNT ****/
#define  CREDIT_CLEARED        "c"    /*** CLEARED IN CREDIT ACCOUNT ***********/
#define  NOT_THERE_YET         "n"    /*** HASN'T REACHED BANK YET *************/
#define  PENDING               "p"    /*** PENDING IN BANK ********************/
#define  LOCKED                "l"    /*** CLEARED, BUT NOT OKAY TO CONDENSE ***/
```

Here is how you should think about transaction status codes.

CREDIT_NOT_THERE_YET	Say you just made a purchase with a credit card, but the transaction has not appeared yet on your credit company's website yet. Temporarily label transactions like these as such until the credit company thinks that transaction is "pending" or cleared .
CREDIT_PENDING	Transactions shown as "pending" in your account on your credit company's website.
NOT_THERE_YET	Say you wrote a check to a friend, but he's lazy and doesn't deposit it for two weeks. Transactions like these, which you know you made but do not appear on the webpage of your bank account yet, should be labeled as such.
PENDING	Formally listed as "pending" on your bank account website.
LOCKED	Say you want to condense your ledger, but you do not want to lose the information for a particular transaction or set of transactions. Label these transactions as locked.

5   Row and column separators

By default, the program expects the ledger file to have tab-delimited columns and rows delimited by either the newline character or the carriage return character. The user can change these settings at

```
#define ROW_SEPARATORS       "\n\r"
#define COLUMN_SEPARATORS     "\t"
```

For example, If I change COLUMN_SEPARATORS to the comma,

```
#define ROW_SEPARATORS       "\n\r"
#define COLUMN_SEPARATORS     ","
```

then the program would expect a CSV (comma-separated values) file as input. These characters should only be used to separate fields. They should not be used in the actual entries of the ledger as content.

6   Column order

The ordering of the columns in the ledger file is defined in the section of include/user_settings.h below.

```
#define AMOUNT        0     /*** COLUMN FOR TRANSACTION AMOUNTS ************/
#define STATUS        1     /*** COLUMN FOR TRANSACTION STATUS CODES *******/
#define CREDIT        2     /*** COLUMN FOR NAMES OF CREDIT ACCOUNTS *******/
#define BANK          3     /*** COLUMN FOR NAMES OF BANK ACCOUNTS *********/
#define PARTITION     4     /*** COLUMN FOR NAMES OF BANK PARTITIONS *******/
#define DESCRIPTION   5     /*** COLUMN FOR DESCRIPTIONS OF TRANSACTIONS ***/
```

The program would then expect a ledger file of the following form:

amount	status	credit	bank	partition	description
−30.14	cn	card1	checking	food	groceries 12/3/13
−15.36	cn	card2	checking	food	produce 12/2/13
−10.41	cp	card1	checking	food	dinner 11/21/13
−7.31	c	card2	checking	fun	movie 11/18/13
−500	p		checking	bills	December rent
−30	l		checking	bills	November electric
−50	l		checking	bills	November heat
−45			checking	bills	The Economist
800	p		checking	rent	November paycheck
400	p		checking	food	November paycheck
500	p		checking	upkeep	November paycheck
50	p		checking	fun	November paycheck
50			checking	fun	won $50 in poker
200			savings	bike	gift money
200			savings	bike	October paycheck
100			savings		general savings
200					cash

which is the current default. But if I were to switch the order of the values of STATUS and AMOUNT,

```
#define AMOUNT       1    /*** COLUMN FOR TRANSACTION AMOUNTS  ************/
#define STATUS       0    /*** COLUMN FOR TRANSACTION STATUS CODES  *******/
#define CREDIT       2    /*** COLUMN FOR NAMES OF CREDIT ACCOUNTS  *******/
#define BANK         3    /*** COLUMN FOR NAMES OF BANK ACCOUNTS  *********/
#define PARTITION    4    /*** COLUMN FOR NAMES OF BANK PARTITIONS  *******/
#define DESCRIPTION  5    /*** COLUMN FOR DESCRIPTIONS OF TRANSACTIONS  ***/
```

Then the program would expect a ledger file of the form,

status	amount	credit	bank	partition	description
cn	−30.14	card1	checking	food	groceries 12/3/13
cn	−15.36	card2	checking	food	produce 12/2/13
cp	−10.41	card1	checking	food	dinner 11/21/13
c	−7.31	card2	checking	fun	movie 11/18/13
	−500		checking	bills	December rent
l	−30		checking	bills	November electric
l	−50		checking	bills	November heat
	−45		checking	bills	The Economist
p	800		checking	rent	November paycheck
p	400		checking	food	November paycheck
p	500		checking	upkeep	November paycheck
p	50		checking	fun	November paycheck
	50		checking	fun	won $50 in poker
	200		savings	bike	gift money
	200		savings	bike	October paycheck
	100		savings		general savings
	200				cash

7   Summary output options

The user can tweak some of the macros in `user_settings.h` to customize the output of the summaries.

```
#define  PRINT_ZEROED_ACCOUNTS     0

#define  USE_COLOR       1

#define  NORMAL_COLOR        "\x1B[0m"     /*** REGULAR TEXT ******/
#define  NEGATIVE_COLOR      "\x1B[31m"    /*** NEGATIVE TOTALS ***/
#define  POSITIVE_COLOR      "\x1B[32m"    /*** POSITIVE TOTALS ***/
#define  ZERO_COLOR          "\x1B[34m"    /*** EMPTY TOTALS ******/
```

PRINT_ZEROED_ACCOUNTS	Set to 1 to print out all accounts, including those with balances of $0.00. Set to 0 to ignore all empty accounts.
USE_COLOR	Set to 1 to print out summaries in color. Set to 0 otherwise. Coloring is best for viewing summaries in the terminal window, but if you want to pipe the output to a file and view it there, then the color codes create extra annoying characters.
NORMAL_COLOR	Shell color code for regular text. The default is white.
POSITIVE_COLOR	Shell color code for positive totals in the summaries. The current default is green.
NEGATIVE_COLOR	Shell color code for negative totals. The current default is red.
ZERO_COLOR	Shell color code for zero totals. The current default is blue.