RemindMe

Michael K K Montague

Table of Contents

1 Introduction

RemindMe is used to keep track of dates. I will start with an example reminder file.

```
;;
;; Example Reminders
;;

jan 1 remind [New Year's Day].
second sun may remind [Mother's Day].
tue on or after nov 2 remind [Election Day].
4 aug 1966 every year remind [My Birthday].
19 may 2005 previous week remind [A Final Exam].
10 may 2005 every week for 10 weeks remind [Swimming Lessons].
12 may and 19 may remind [Final Exam].
```

These lines are comments; everything after '`;`' on a line is ignored.

```
;;
;; Example Reminders
;;
```

January 1st every year is New Year's day. '`jan 1`' is the date that we are matching; it also could have been written '`1 jan`'. There is no year specified. That means that it will match January 1st every year. '`remind`' is the action we want to happen when there is a match and '`New Year's Day`' is the message. Finally, the statement must be terminated with a period ('`.`').

```
jan 1 remind [New Year's Day].
```

The second Sunday in May is Mother's day. The '`first`', '`second`', '`third`', '`fourth`', '`fifth`', and '`last`' day of a month can be specified. Both the month and year are optional.

```
second sun may remind [Mother's Day].
```

In the USA, the Tuesday on or after November 2nd is election day. This could have also been specified as '`tue after nov 1`'.

```
tue on or after nov 2 remind [Election Day].
```

Sometimes it is nice to have a base date and then to be reminded at a regular interval. My birthday is a good example of this. Follow a date by '`every`', an optional number, and '`day`', '`week`', '`month`', or '`year`'.

```
4 aug 1966 every year remind [My Birthday].
```

To get advanced notification of an event, use '`previous`' followed by an optional number and '`day`' or '`week`'. This says to remind you in advance of the event.

```
19 may 2005 previous week remind [A Final Exam].
```

To specifying a repeating event which goes on for a fixed amount of time use '`every`' and '`for`'. To repeat until a specific date, use '`until`'.

```
10 may 2005 every week for 10 weeks remind [Swimming Lessons].
```

If you want to have multiple dates for a single action, use '`and`'.

```
12 may and 19 may and first mon jun remind [Final Exam].
```

Finally, to run the example reminder file, use '`runrme`'. The following command will work if the file is called '`example.rme`'.

```
runrme example.rme
```

2 Reminders

A reminder file consists of statements. Each statement is a date specifier followed by an action and terminated with a period. Each date specifier may optionally have a previous clause: this says to remind about this event ahead of time.

2.1 Dates

A date contains four parts: a day of the month, a day of the week, a month, and a year. The parts can be specified in any order and they are all optional. The date must be valid.

- '`1 jan 1982`': specifies 1 January 1982.
- '`1 jan`': specifies 1 January of every year.
- '`1`': specifies the first day of every month in every year.
- '`jan`': specifies every day in January in every year.
- '`1982`': specifies every day in 1982.
- : specifies every day of every month of every year.
- '`feb 31`': *invalid* date: February has at most 29 days.
- '`fri`': specifies all Fridays.
- '`fri jan`': specifies all Fridays in January every year.
- '`fri 1`': *invalid* date: the first day of every month is not always a friday.
- '`fri 1 jan`': *invalid* date: January 1 is not always a Friday.
- '`fri 1 jan 1982`': specifies 1 January 1982, which is a Friday.

2.2 Nth Week Day

2.3 Offset

2.4 Every

2.5 And

2.6 Previous

2.7 Actions

3 Examples

April Fool's day is April 1st.

```
apr 1 remind [April Fool's Day].
april 1 remind [April Fool's Day].
1 apr remind [April Fool's Day].
```

The third Monday in January is Martin Luther King Day.

```
third mon jan remind [Martin Luther King Day].
third monday january remind [Martin Luther King Day].
```

In the USA, the Tuesday after November 1st is election day.

```
tue after nov 1 remind [Election Day].
tuesday after 1 nov remind [Election Day].
tue on or after nov 2 remind [Election Day].
```

In the USA, the first week day on or after April 15th is when income taxes are due.

```
mon tue wed thu fri on or after april 15 remind [Income Taxes Due].
```

My book club is the second Wednesday of every month.

```
second wednesday remind [Book Club].
```

My birthday is the 4th of August.

```
4 aug remind [My Birthday].
4 aug 1966 every year remind [My Birthday %(age)].
```

I have a final exam on May 19th and June 1st.

```
may 19 and jun 1 remind [Final Exam].
```

4 Reference

4.1 Specifying Dates

A specifier specifies on which dates the action should occur.

```
<specifier> := <date-specifier> [and <date-specifier>] ... [<previous>]
<date-specifier> := <date>
        | <nth-weekday> <date>
        | <offset> <date>
        | <date> <every>
        | <specifier-test>
        | <specifier-set>
```

A date consists of zero or more of a day, a day name, a month, and a year. Each may be specified no more than once in any order. If day and day name are both specified, then month and year must be specified *and* the day and day name must not conflict.

```
<date> := [<day> | <dayname> | <month> | <year>] ...
<day> := 1 | 2 | ... | 31
<dayname> := mon | tue | ... | sun
        | monday | tuesday | ... | sunday
<month> := jan | feb | ... | dec
        | january | february | ... | december
<year> := 1800 | ... | 3000
```

The nth weekday is specified using one of first, second, etc. and a weekday. This will select dates which are the nth weekday of a month.

```
<nth-weekday> := <nth> <weekday>
<nth> := first | second | third | fourth | fifth | last
```

An offset is used to shift a date by some amount. The shift can be forward (after) or backwards (before). The amount is in weekdays.

```
<offset> := <weekday> <weekday> ... [on or] before
        | <weekday> <weekday> ... [on or] after
```

Every is used to repeat a date at some regular interval. It can go on forever, stop after a certain number of times, or stop after a particular date. Finally, particular dates can be excluded using except.

```
<every> := every [<n>] <step> [<done>] [<except> ...]
<step> := day[s] | week[s] | month[s] | year[s]
<done> := until <date>
        | for <n> <step>
<except> := except <date>
```

Previous is used to remind before an event.

```
<previous> := previous [<n>] [day | days | week | weeks]
<n> := 1 | 2 | ...
```

4.2 Running Reminder Files

'`runrme`' is used to run a reminder file. Without any arguments, '`runrme`' will try to run a reminder file in a default location for today.

```
runrme [option] [date]
    -b days
    -a days
    -f filename
    date: day, month, and year in any order
    month is jan, feb, ... dec
```

date specifies what date to use in running the reminder file; it defaults to today. '`runrme`' processes the reminder file for the date and then some number of days before and after. '`-b days`' specifies the number of days before; it defaults to zero. '`-a days`' specifies the number of days after; it defaults to zero.

'`-f filename`' specifies the reminder file to be used.

On Windows, the default filenames tried are as follows.

```
%USERPROFILE%\remindme.rme
%USERPROFILE%\remindme\remindme.rme
My Documents\remindme.rme
My Documents\RemindMe\remindme.rme
```

On Unix and Mac OS X, the default filenames tried are as follows.

```
~/.remindme
~/remindme.rme
~/.remindme/remindme.rme
~/remindme/remindme.rme
~/Documents/remindme.rme
~/Documents/RemindMe/remindme.rme
```

4.3 Specifying Actions

```
<action> := remind <string>
         | evaluate <expression>
         | eval <expression>
<string> := [ ... ]
```

Use `\]` to specify a `]` in a string and `\\` to specify a `\`.

4.4 Using Tiny Scheme

`date` is a TSCM type.

`date?` *obj* [Procedure]

`date?` returns `#t` if *obj* is a pair, and otherwise returns `#f`.

`date` *day month year* [Procedure]

Create a new date having the given *day*, *month*, and *year*.

`day` *date* [Procedure]
`month` *date* [Procedure]
`year` *date* [Procedure]
Get the day, month, and year parts of a date.

`day-name` *date* [Procedure]
Get the day name for a particular date.

```
(day-name (date 28 4 2005))
    ⇒ thursday
```

`days-between` *date1 date2* [Procedure]
Return the number of days between *date1* and *date2*.

```
(days-between (date 28 4 2005) (date 20 4 2005))
    ⇒ 8
```

`date+` *date n* [Procedure]
`date-` *date n* [Procedure]
Add or subtract *n* days to the *date* and return the resulting date.

`test-specifier` *proc name* [Procedure]
`test-specifier` makes *proc* be the specifier for *name*. The return value of `test-specifier` is unspecified.

The *proc* must be a procedure which takes a single argument, a date, and return `#f` if the date does not match.

```
(define (friday-the-13th d)
    (if (and (= (day d) 13) (eq? (day-name d) 'friday))
        #t
        #f))
(test-specifier friday-the-13th "friday-the-13th")
```

`set-specifier` *proc name* [Procedure]
`set-specifier` makes *proc* be the specifier for *name*. The return value of `set-specifier` is unspecified.

The *proc* must be a procedure which takes a single argument, a year, and return a list of zero or more dates which match for the year. Return the empty list, `'()`, if no dates match.

```
(define (friday-the-13th y)
    (define (generate m y)
        (if (> m 12)
            '()
            (if (eq? (day-name (date 13 m y)) 'friday)
                (cons (date 13 m y) (generate (+ m 1) y))
                (generate (+ m 1) y))))
    (generate 1 y))
(set-specifier friday-the-13th "friday-the-13th")
```

`remind-field` *proc name* [Procedure]

`remind-field` makes *proc* by the field for *name*. The return value of `remind-field` is unspecified.

The *proc* must be a procedure which takes three arguments: the current date, the base date, and the match date. Using the Tiny Scheme procedure, `display`, the result of *proc* replaces the field in the output.

The current date is the date that RemindMe is currently using to match against possible reminders. When RemindMe is run on a particular date, typically today, the current date is that particular date. When RemindMe is run to generate a calendar, say for a month, the current date will be each day of the month as RemindMe walks through the days of the month.

The base date is the date specified in the reminder. In the following examples the current date is '`12 apr 2004`'.

```
(define (base-date cd bd md)
    bd)
(remind-field base-date "base-date")

12 apr 1966 every year remind [%(base-date)].
    ⇒ #<date: 12 apr 1966>
12 may 1966 every month remind [%(base-date)].
    ⇒ #<date: 12 apr 1966>
12 jan every month remind [%(base-date)].
    ⇒ #<date: 12 jan 2004>
```

The match date is date which actually matched the reminder. If there is no '`previous`' then the match date and the current date will always be the same. If there is a '`previous`' then the match date might be after the current date. As before, in the following examples the current date is '`12 apr 2004`'.

```
(define (current-match cd bd md)
    (list cd md))
(remind-field current-match "current-match")

12 apr 2004 remind [%(current-match)].
    ⇒ (#<date: 12 apr 2004> #<date: 12 apr 2004>)
16 apr 2004 previous 10 days remind [%(current-match)].
    ⇒ (#<date: 12 apr 2004> #<date: 16 apr 2004>)
```

Here is a useful example. Say you want to keep track of a reminder for someone's birthday as well as when they were born.

```
(define (age cd bd md)
    (- (year cd) (year bd)))
(remind-field age "age")

4 aug 1966 every year
        remind [Michael's Birthday; he is %(age) years old.].
    ⇒ Michael's Birthday; he is 38 years old.
```

`include` *filename* [Procedure]

Include the file named by *filename* in the current script at the current point.