

# **Remind: The UNIX Geek's Ultimate Calendar**

**Dianne Skoll  
Roaring Penguin Software**

**6 November 2007**

# Introduction

- What's wrong with most calendar tools
- What Remind is and is not
- History of Remind
- Simple reminders
- Holidays and exceptions
- Advanced Remind: Expressions
- Front-ends and Back-ends

# Most Calendar Tools

- Are graphical (== slow and bloated.)
- Store data in a non-human-readable format (require import/export.)
- Don't work well over SSH connections.
- Are unsophisticated (can't handle simple things like moving garbage days easily.)
- Do not follow the UNIX philosophy: Small tools that do their jobs well and communicate simply over pipes.

# What Remind Is

- UNIX command-line tool that reads a text file for its database.
- Sophisticated tool for date calculation.
- Scripting-language interpreter.
- Small (120kB executable) and *fast*.
- Easy to learn.
- Flexible.
- Free (GPL).

# What Remind Is Not

- Shared calendar. It's meant for personal use.
- Standards-compliant. There are ical converters in and out, but Remind is *far* more flexible than any other calendar tool and can express reminders that are impossible to express except in Remind!
- GNOME/KDE/DBUS/Your-Buzzword-Here compliant.

# History of Remind

- Early 1989: dfs got first taste of UNIX.
- Late 1989: dfs stuck on MS-DOS.
  - “I need a **calendar** replacement!”
  - Wrote remind v1.0 (never released).
- 1990: Worked at Carleton as research engineer. Wrote Remind 2.0 for fun (SunOS). Released on alt.sources.
- 1992: Wrote Remind 3.0 to avoid working on Master's thesis. Released on comp.sources.misc.

# History of Remind - 2

- Continued working on Remind in fits and starts. More or less stopped in November 2005.
- *Wham!* A bunch of Mac OS X users discover Remind after it is featured on 43folders.com. “Wow!” they say. “This command-line stuff is *powerful!*”
- Picked Remind development up again in July 2007 and rejoined the mailing list (which I thought had gone moribund...)

# Running Remind

- Use your favourite text editor (emacs or... that other one...) to create a reminder file.
- This file contains commands in Remind's scripting language.
- Then invoke Remind:
- `remind name_of_file`



# A Reminder File

- What does a Reminder file look like?  
Well, it could be as simple as this:

```
# Here's a comment.
```

```
REM 6 January MSG Dianne's Birthday
```

```
REM 6 Nov 2007 MSG OCLUG Talk
```

```
REM Wed AT 16:30 MSG E's ballet
```

# A Reminder File - 2

- Or it could be as nasty as this:

```
FSET isGood(date) \  
    monnum(mondate(2, date)) == \  
    monnum(mondate(2, mondate(2, date)+1))
```

```
REM 1 SATISFY isGood(trigdate())
```

```
set x mondate(2, mondate(2, trigdate()+1))
```

```
REM MSG [x] - This will be explained.
```

# How Remind Works

- Everyone uncurl from the fetal position after that last slide...
- Remind works like this:
  - It reads a list of commands from the reminder file.
  - It executes the commands.
  - Each *line* is a command... \ but there is backslash-continuation.
  - Lines starting with **#** or **;** and blank lines are ignored.

# The REM Command

- The REM command is used to issue reminders.
- The Remind algorithm:
  - Each REM command has a *date specification*.
  - Remind starts **from** today and checks whether or not the date specification is satisfied. If not, it increments the date and checks again.
  - Remind keeps going until it finds a date that satisfies the spec (the *trigger* date) or proves to itself that no such date exists.

# The REM Command

- Actually, I lied. Remind does not use the algorithm on the previous slide; it would be much too slow.
- However, it *behaves* as if it uses the previous algorithm, and understanding and accepting that is key to **Remind Zen**. *If you don't get the trigger algorithm, you won't get Remind.*
- Actually, I'm lying again... Remind has features that let you modify the algorithm a bit. But never mind that for now...

# A REM Dissected

- Take a look at:  
**REM 6 Jan MSG Dianne's Birthday**
- The date spec **6 Jan** contains two components: A day number and a month name.
- This reminder is triggered every 6<sup>th</sup> of January, regardless of the year.
- Quiz: On 6 November 2007, what is the *trigger date* of the reminder above?
- Answer: 6 January 2008.

# More Dissection

- And:  
**REM 6 Nov 2007 MSG OCLUG Talk**
- The date spec **6 Nov 2007** contains three components: A day number, a month name and a year.
- This reminder is triggered once only: On 6 November 2007. No other date *satisfies* the date specification.
- Quiz: On 7 November 2007, what is the trigger date of the reminder above?
- Answer: *None*. No trigger can be computed.

# Quick Dissections

- REM Wednesday MSG Every Wednesday
- REM Mon 1 MSG First Monday on or after the first of the month
- REM Mon Tue Wed Thu Fri MSG Every weekday
- REM Mon Tue Wed Thu Fri 1 Feb MSG First weekday of February in every year
- REM March 2008 MSG Every day in March 2008



# Weekday Names

- Weekday names in a date spec are special.
- If there is *no* day number, then the reminder triggers on all the named weekdays, subject to other constraints.
- If there *is* a day number, then the reminder triggers *only* on the *first* day on or after the day number that is *also* in the list of day names.
- Reread the above point until you understand...

# Weekday Names

- So what does this do:

```
REM Tue 8 MSG Something...
```

- Answer: Triggers on the second Tuesday of the month.
- Huh??
- Well, the first Tuesday on or after the 8<sup>th</sup> of the month is the second Tuesday of the month.

# Advance Warning

```
REM 6 Jan +5 MSG %"Eleanor's \
Anniversary%" is %b.
```

- This reminder is triggered on 6 January *and* on the five preceding days.
- The sequence “%b” is replaced with “in 3 days' time”, “tomorrow” or “today” as appropriate.
- The magic sequence %” will be explained later.

# Backward Scanning

- We've seen how to do the first, second etc weekday of a month. How about the *last* weekday of a month? That's a bit trickier...

REM Mon 1 --7 MSG Last Monday \  
of the month.

- The “Mon 1” means “First Monday on or after the 1<sup>st</sup> of the month.”
- The “--7” means “... and then go back 7 days.”

# Backward Scanning

- Question: How would you trigger a reminder on the last Friday of every June?
- Answer:  
`REM Fri 1 July --7 MSG Wookiee!`

# Non-Weekly Repeats

```
REM 7 Sep 2007 *14 MSG \  
Every 2nd Friday.
```

```
REM 10 Mar 2008 *1 UNTIL 14 Mar 2008 \  
MSG March Break
```

- $*n$  means repeat every  $n$  days. However, you must supply day, month and year of start date.
- UNTIL clause limits ending date of repeat.

# The OMIT Context

- Our lovely regular reminders are often thrown off by nasty... holidays.
- For example, we have a sales meeting every Monday, but it's moved to a Tuesday if there's a holiday.
- First, let's look at how to tell Remind about holidays.

# The OMIT Context

- Remind keeps a list of *OMITs*. These are dates that are handled specially.

- OMITs can be *full* or *partial*. Here's a full OMIT:

**OMIT 3 Sep 2007**

- And here's a partial one:

**OMIT 25 Dec**

- You can include a MSG for convenience:

**OMIT 3 Sep 2007 MSG Labour Day**

**OMIT 25 Dec MSG Christmas**



# How OMITs are Handled

- OMITs affect advanced reminders and backward scanning. Read carefully:
- An advance warning of  $+n$  or a backward scan of  $-n$  does **not** count OMITted days.
- An advance warning of  $++n$  or a backward scan of  $--n$  **does** count OMITted days.
- Examples on next slide:

# OMITs in Action

OMIT 25 Dec MSG Christmas

REM 26 Dec -1 MSG Triggered on 24 Dec

REM 26 Dec --1 MSG Triggered on 25 Dec

REM 26 Dec +1 MSG Start warning 24 Dec \  
for %y-%m-%d

REM 26 Dec ++1 MSG Start warning 25 Dec \  
for %y-%m-%d

# REM vs OMIT

- If a reminder falls on an OMITted day, Remind normally triggers it. You can change the behaviour with BEFORE, AFTER and SKIP. BEFORE moves the reminder before any OMITted days. AFTER moves it after, and SKIP simply skips it.
- Examples coming up...

# REM vs OMIT Examples

# Christmas is a Tuesday in 2007

OMIT 25 Dec MSG Christmas

OMIT 26 Dec MSG Boxing Day

REM Wed MSG Triggered on 26<sup>th</sup> anyway

REM Wed BEFORE MSG Triggered on 24<sup>th</sup>

REM Wed AFTER MSG Triggered on 27<sup>th</sup>

REM Wed SKIP MSG Skipped on 26th

# Saving the OMIT Context

- You can save the OMIT context with PUSH-OMIT-CONTEXT.
- Restore with POP-OMIT-CONTEXT.
- Clear with CLEAR-OMIT-CONTEXT.

**PUSH-OMIT-CONTEXT**

**CLEAR-OMIT-CONTEXT**

**# Add a bunch of exceptions**

**OMIT ... whatever ...**

**OMIT ... whatever ...**

**REM ... SKIP MSG whatever**

**POP-OMIT-CONTEXT**

# Local OMITs

- The REM command has an OMIT clause that lets you add *weekday names* to the OMIT context temporarily.
- For example, this reminder triggers on the last weekday of every month:

```
REM 1 OMIT Sat Sun -1 MSG \  
Last weekday of the month.
```

# Timed Reminders

- The AT clause introduces a timed reminder. Remind can run in the background and pop up warnings. Example:

```
REM Tuesday AT 18:00 +60 *10 MSG Ballet
```

- Every Tuesday at 6:00pm, this reminder triggers. In addition, it nags me every 10 minutes starting 60 minutes ahead of 6:00pm.

# Variables and Expressions

- You can store values in variables. A variable name follows C naming conventions.
- Variables are not typed; they take on the type of whatever you store in them.
- Remind expressions resemble C expressions:

```
SET a 1 + 2 * 3  
# Sets a to 7
```



# Data Types

- INTs: Integers. Examples: 0, 1, 42, -39
- STRINGs: "foo", "test", "wookie"
- DATEs: '2007-11-06', '1999-09-01'
- TIMEs: 14:23, 9:00
- DATETIMEs: '2007-11-06@19:00'

# Sample Expressions

- `1 + 2 + "foo"` `"3foo"`
- `1 + (2 + "foo")` `"12foo"`
- `'2007-11-06' - 23` `'2007-10-14'`
- `14:30 - 4:45` `585`
- `'2007-11-06@19:00' - 60*36`  
`'2007-11-05@07:00'`

# Built-in Functions

- Remind has *many* built-in functions.
  - String manipulation
  - Arithmetic (abs, max, min, etc.)
  - Selectors (iif, choose)
  - Type conversion (coerce)
  - Date/time (current, today, year, monnum)
  - Specialized functions (easterdate, hebdate)
  - Astronomical functions (sunrise, sunset, moondate, moontime)

# Built-in Function Examples

- `max(42, 17, 199)`  
`199`
- `moondate(2, '2007-11-06')`  
`'2007-11-24'`
- `easterdate('2008-01-01')`  
`'2008-03-23'`
- `sunset('2007-11-06')`  
`16:44` (*in Ottawa*)
- `trigger('2007-11-06')`  
`"6 November 2007"`

# Expression-Pasting

- Anywhere Remind is parsing a line, you can *paste in* an expression.
- Surround the expression with square brackets.
- The following two sequences are equivalent:

```
REM 24 Nov MSG Hello!
```

```
set a "Hello"
```

```
set b 26
```

```
REM [b-2] Nov MSG [a]!
```

# Expression-Pasting (2)

- Examples:

```
REM [trigger(easterdate())] MSG Easter Sunday
```

```
REM [trigger(easterdate()-2)] MSG Good Friday
```

```
REM 6 JAN MSG \
```

```
Dianne's [ord(year(trigdate())-1967)] Birthday
```

```
REM [trigger(moondate(2))] MSG Full Moon
```

```
REM [trigger(hebdate(1, "Tishrey"))] \
```

```
MSG Rosh Hashana
```

# The Substitution Filter

- Before issuing a reminder, Remind passes the body through a filter.
- It replaces various %-sequences with useful things. For example:  
**REM 6 Nov 2007 +2 MSG OCLUG %b.**  
prints the following:  
4 Nov 2007:       **OCLUG in 2 days' time.**  
5 Nov 2007:       **OCLUG tomorrow.**  
6 Nov 2007:       **OCLUG today.**
- Consult man page for all the %-sequences.

# Back-Ends

- Remind can be invoked with a **-p** option. This outputs Reminders in a computer-parseable format.
- Back-ends can munge the output. Remind ships with three back-ends:
  - rem2ps: Create PostScript calendars.
  - tkremind: Graphical front/back-end.
  - rem2html: Create HTML calendars.
- There are third-party back-ends (wyrd, wxremind) as well.




# Out-of-Band Reminders

- Remind has a way to pass special data to back-ends. This is back-end specific, but my back-ends support:
  - Shading days in the calendar.
  - Drawing coloured reminders.
  - Drawing moon-phase indicators.
- rem2ps allows insertion of PostScript.
- rem2html allows insertion of HTML.
- Can do all kinds of fancy things! Read man page for details.

# rem2ps Screenshot

November 2007

Sunday	Monday	Tuesday	Wednesday	Thursday
<p>October</p> <p>S M T W T F S</p> <p>1 2 3 4 5 6</p> <p>7 8 9 10 11 12 13</p> <p>14 15 16 17 18 19 20</p> <p>21 22 23 24 25 26 27</p> <p>28 29 30 31</p>	<p>December</p> <p>S M T W T F S</p> <p>1</p> <p>2 3 4 5 6 7 8</p> <p>9 10 11 12 13 14 15</p> <p>16 17 18 19 20 21 22</p> <p>23 24 25 26 27 28 29</p> <p>30 31</p>			 <p>Dean</p> <p>20 Hes</p>
<p>4</p> <p>9:00am Hebrew</p> <p>23 Heshvan</p>	<p>5</p> <p>5:00pm Ballet Piano</p> <p>24 Heshvan</p>	<p>6</p> <p>6:00pm Stages 7:00pm OCLUG Meeting Dad's birthday</p> <p>25 Heshvan</p>	<p>7</p> <p>4:30pm Hebrew 4:30pm Ballet</p> <p>26 Heshvan</p>	<p>27 Hes</p>
<p>11</p> <p>9:00am Hebrew</p>	<p>12</p> <p>5:00pm Ballet Piano</p>	<p>13</p> <p>6:00pm Stages</p>	<p>14</p> <p>4:30pm Hebrew 4:30pm Ballet</p>	

# tkremind Screenshot

November 2007 - TkRemind

November 2007						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1 17:20 Dean's 23rd birthday	2 6:45pm Ballet Alan's birthday Black Box	3 9:00am Ballet
4 9:00am Hebrew	5 5:00pm Ballet Piano	6 6:00pm Stages 7:00pm OCLUG Meeting Dad's birthday	7 4:30pm Hebrew 4:30pm Ballet	8	9 18:04 6:45pm Ballet Blue Box	10 9:00am Ballet
11 9:00am Hebrew	12 5:00pm Ballet Piano	13 6:00pm Stages	14 4:30pm Hebrew 4:30pm Ballet	15	16 6:45pm Ballet Black Box	17 17:33 9:00am Ballet
18 9:00am Hebrew	19 5:00pm Ballet Piano	20 6:00pm Stages	21 4:30pm Hebrew 4:30pm Ballet	22	23 6:45pm Ballet Eleanor's birthday Blue Box	24 09:31 9:00am Ballet
25 9:00am Hebrew	26 5:00pm Ballet Richard Bartholomew's birthday Piano	27 6:00pm Stages	28 4:30pm Hebrew 4:30pm Ballet Lisa's birthday	29	30 6:45pm Ballet Black Box GST/HST Remittance	

<- Today -> Go To Date... Print... Options... Quit 25 Sep 2007 06:26PM 0 reminders queued

# Explanation of Nasty File

- A **Blue Moon** is the second full moon in a calendar month. Relatively rare...

```
FSET isFirstFull(date) \  
monnum(mondate(2, date)) == \  
monnum(mondate(2, mondate(2, date)+1))
```

- isFirstFull is a *user-defined* function. If the month-number of the *next* full moon after *date* is the same as the month-number of the full moon *after that full moon*, return true. Otherwise, return false.

# Explanation (2)

```
REM 1 SATISFY isFirstFull(trigdate())
```

- Step through the 1<sup>st</sup> of every month until you find one where the next full moon is the same as the full moon after the next one. Then stop.

```
SET blue moondate(2, moondate(trigdate())+1)
```

- Set blue to the date of the 2<sup>nd</sup> full moon. Then print the answer:

```
REM MSG Next blue moon is [trigger(blue)]
```

```
Next blue moon is 31 December 2009
```

# Quick Demos

- (Demo building Remind.)
- (Demo tkremind, wyrd, wxremind, etc.)

# The Tip of the Iceberg

- We have covered just the tip.
- For the rest of the iceberg, visit:

<https://dianne.skoll.ca/projects/remind>

- Download Remind, build and install, read the man page, and have fun!
- Thank you!
- Questions?