mcron - Mellor's cron daemon

Dale Mellor

Table of Contents

1 Introducing mcron

The mcron program represents a complete re-think of the cron concept originally found in V7 Unix, ratified by POSIX, and re-realized by Paul Vixie for the free world. The original idea was to have a daemon that wakes up every minute, scans a set of files under a special directory, and determines from those files if any shell commands should be executed in this minute.

The new[1] idea is to read the required command instructions, work out which command needs to be executed next, and then sleep until the inferred time has arrived. On waking the commands are run, and the time of the next command is computed. Furthermore, the specifications are written in Scheme, allowing at the same time simple command execution instructions and very much more flexible ones to be composed than the POSIX format allows. This has several useful advantages over the original idea.

- Does not consume CPU resources when not needed. Many cron daemons only run jobs once an hour, or even just once a day. Busy computers with high memory pressure can keep the mcron program swapped out to disk more of the time.
- Can easily allow for finer time-points to be specified, i.e. seconds. In principle this could be extended to microseconds, but this is not implemented.
- Times can be more or less regular. For example, a job that runs every 17 hours can be specified, or a job that runs on the first Sunday of every month.
- Times can be dynamic. Arbitrary Guile (Scheme) code can be provided to compute the next time that a command needs to be run. This could, for example, take the system load into consideration.
- Turns out to be easy to provide complete backwards compatibility with the POSIX cron specification.
- Each user looks after their own files in their own directory. They can use more than one to break up complicated cron specifications.
- Each user can run their own daemon. This removes the need for suid programs to manipulate the crontabs, and eliminates many security concerns that surround all existing cron programs.
- The user can obtain an advance schedule of all the jobs that are due to run.
- Vixie cron is implemented in 4500 lines of C code; mcron was 2000 lines of Scheme (it has grown a little fatter since), despite the fact that it offered many more features and much more flexibility, and complete compatibility with Vixie cron and the POSIX specification.

A full discussion of the design and philosophy of mcron can be found in the white paper at `http://www.gnu.org/software/mcron/design.html`.

[1] With retrospect, not so new: AT&T and Berkeley Unices had a cron program which also implemented this 'new' idea, though Dale Mellor was unaware of this at the time of mcron's conception and initial development. The Wikipedia page for cron appears quite authoritative and is a good source of information about the history of cron.

2 Simple examples

The vast majority of uses of cron are sublimely simple: run a program every hour, or every day. With this in mind the design of mcron has been to allow such simple specifications to be made easily. The examples show how to create the command descriptions, and subsequently how to run mcron to make them happen.

2.1 Guile

You have an executable `my-program` in your home directory, which you want to run every hour. Create a file `job.guile` in directory `~/.config/cron` (this path may be altered by the `$XDG_CONFIG_HOME` environment variable) with the following contents

```
(job '(next-hour) "my-program")
```

then run the command `mcron`.

Want the program to run fifteen minutes past the hour, every two hours? Edit the file to read

```
(job
   '(next-minute-from
      (next-hour (range 0 24 2))
      '(15))
   "my-program")
```

and run the command `mcron`.

Or, if you are not comfortable with Scheme, you could use (and see also the next section)

```
(job "15 */2 * * *" "my-program")
```

and run the `mcron` command.

If you want to run other jobs, you can either add more lines to this file, or you can create other files in your `.config/cron` directory with the `.guile` extension. Alternatively, you can use any file you want and pass it as an argument to `mcron`, or even pipe the commands into the standard input.

2.2 POSIX

You have an executable `my-program` in your home directory, which you want to run every hour. Create a file `job.vixie` in directory `~/.cron` with the following contents

```
0 * * * * my-program
```

then run the command `mcron`.

Alternatively (full compatibility with Vixie cron), set your environment variable `EDITOR` to your favorite editor, run `crontab -e`, put the above line into the edit buffer, save and exit. For this to work the `cron` daemon must be already running on your system, as root. This style of operations is considered deprecated.

3 Full available syntax

3.1 Guile Syntax

3.1.1 Job specification

In Guile-formatted configuration files each command that needs executing is introduced with the `job` function. This function takes two mandatory arguments, the first a time specification, and the second a command specification. An optional third argument may contain a string to display when this job is listed in a schedule. Additionally a *user* keyword argument can be supplied to use a different user than the one defined in the `configuration-user` global variable.

The first argument can be a procedure, a list, or a string. If a function is supplied, it must take exactly one argument, which will be the "current" time in UNIX format, and the return value of the function must be the time in UNIX format when this action should next be run. The following functions are available to facilitate the computation:

`(next-second-from time . args)` without arguments this returns the second after the given `time`. If an extra argument is supplied, it should form a list of seconds in the minute when the action should run, and the function will return the time of the next allowed second (which may be in the next minute of the hour).[1]

Similarly to `next-second-from`, there are also `next-minute-from`, `next-hour-from`, `next-day-from`, `next-month-from`, `next-year-from`.

Furthermore, the optional argument can be fulfilled by the Guile function `(iota count . start step)`, which will provide a list of `count` values from `start`, with the step if given. For example `(iota 10 0 2)` will yield the list `'(0 2 4 6 8)`.

As a convenience, mcron itself provides a function `range`, such that `(range start end . step)` will provide a list of values from start to (but not including) end, with the step if given. For example `(range 0 10 2)` will also yield the list `'(0 2 4 6 8)`.

If the first argument to the `job` function is a list, it is taken to be program code (technically known as an *S-expression*) made up of the functions `(next-second . args)`, `(next-minute...)`, etc, where the optional arguments can be supplied with the `iota` or `range` functions above (these functions are analogous to the ones above except that they implicitly assume the current time; it is supplied by mcron when the list is *eval*'d).

If the first argument to the `job` function is a string, it is expected to be a POSIX crontab-style time specification. See the section on Vixie syntax for this.

The second argument to the `job` function can be either a string, a list, or a function. The command is executed in the home directory and with the UID of *user*. If a string is passed, it is assumed to be shell script and is executed with the user's default shell. If a list is passed it is assumed to be Scheme code and is *eval*'d as such. A supplied function should take exactly zero arguments, and will be called at the pertinent times.

[1] Note that while commands can be scheduled to run at any second, it is unlikely that they will be executed then but some time shortly thereafter, depending on the load on the system and the number of jobs that mcron has to start at the same time.

3.1.2 Sending output as e-mail

When jobs are specified in a POSIX-style configuration, the command is broken at a percentage sign, and the stuff that comes after this is sent into the command's standard input. Furthermore, any output from the command is mailed to the user. This functionality is provided in mcron for compatibility with Vixie cron, but it is also available to Scheme configuration files. The command (with-mail-out action . user) can be used to direct standard output from the action (which may be a procedure, list, or string) into an e-mail to the user.

In the case that the action is a string, then percentage signs are processed as per the POSIX specifications, and information is piped to the shell command's standard input.

3.1.3 Setting environment variables

Also for compatibility with Vixie cron, mcron has the ability to set environment variables in configuration files. To access this functionality from a Scheme configuration file, use the command (append-environment-mods name value), where name is the name of an environment variable, and value is the value put to it. A value of #f will remove the variable from the environment.

Note that environment modifications are accumulated as the configuration file is processed, so when a job actually runs, its environment will be modified according to the modifications specified before the job specification in the configuration file.

3.2 Extended Guile examples

While Guile gives you flexibility to do anything, and the power to represent complex requirements succinctly, things are not always as they seem. The following examples illustrate some pitfalls, and demonstrate how to code around them.

3.2.1 Synthesizing "at" commands

The current implementation of mcron does not provide for an *at* command (a command-line program that allows the user to specify that a job runs exactly once at a certain time). This can, however, be achieved.

Suppose the program `my-program` needs to be run at midnight tonight. A Guile script like the following would work (but a printed schedule, obtained with the `--schedule` option, will show superfluous entries).

```
(job '(next-day)
     (lambda () (system "my-program")
                (kill (getppid) SIGINT)))
```

3.2.2 Weekly

The astute reader will have noticed that there are no `next-week` or `next-week-from` functions. This is because these concepts are fraught with ambiguity: does "next week" mean seven days from now, or next Sunday, or next Monday, or what? If the month does not start on a week boundary, are the first few days considered the first week, or does the first week begin on the first Sunday (or Monday)? This is important because, for example, trying to specify the second Thursday in a month might actually get the third one if the latter interpretation is held. Because of this we do not provide these functions.

If you really want a job to run every seven days, you could use code like

```
(job (lambda (current-time) (+ current-time (* 7 24 60 60)))
     "my-program")
```

3.2.3 Every second Sunday

To run `my-program` on the second Sunday of every month, a Guile script like the following should suffice (it is left as an exercise to the student to understand how this works!)

```
(job (lambda (current-time)
       (let* ((next-month (next-month-from current-time))
              (first-day (tm:wday (localtime next-month)))
              (second-sunday (if (eqv? first-day 0)
                                 7
                                 (- 14 first-day))))
         (+ next-month (* 24 60 60 second-sunday))))
     "my-program")
```

3.2.4 Two hours every day

Surprisingly perhaps, the following will **not** have the desired effect.

```
(job '(next-hour-from (next-day) '(1 2))
     "my-program")
```

Rather than running the my-program program at one o'clock and two o'clock every day, it will only run it at one o'clock. This is because each time mcron has to compute the next time to run the command, it first obtains the next day, and then finds the earliest hour in that day to run at. Thus, after running the command at one o'clock, the program first skips forwards to the next midnight (missing the two o'clock appointment), and then finds the next one o'clock schedule.

The following simple command is the correct way to specify this behaviour.

```
(job '(next-hour '(1 2)) "my-program")
```

3.2.5 Missing the first appointment

The command

```
(job '(next-hour-from (next-day) '(16))
     "my-program")
```

will run `my-program` every day at four o'clock in the afternoon. However, if mcron is started with this script at midday, the first time the command will run will be four o'clock tomorrow; today's appointment will be missed (one time only).

The correct way to specify this requirement is simply

```
(job '(next-hour '(16))
     "my-program")
```

3.2.6 Penultimate day of every month

The following will run the `my-program` program on the second-to-last day of every month.

```
(job '(- (next-month-from (next-month)) (* 48 3600))
     "my-program")
```

3.3 Vixie

NOTE that this section is definitive. If there is a difference in behaviour between the mcron program and this part of the manual, then there is a bug in the program. This section is also copied verbatim from Paul Vixie's documentation for their cron program, and their copyright notice is duly reproduced below.

There are three problems with this specification.

1. It is allowed to specify days of the month in the range 0-31. What does it mean to specify day 0? Looking at the Vixie source code, it seems that if this date appears as part of a list, it has no effect. However, if it appears on its own, the effect is to say "don't run on any particular day of the month, only take the week-day specification into account." Mcron has been coded to mimic this behaviour as a special case (unmodified mcron logic implies that this date specification would cause jobs to run on the last day of the previous month, which will never happen).

2. Similarly to the above (but different), months of the year can be specified in the range 0-12. In the case of mcron (don't know what Vixie cron did) month 12 will cause the program to wait until January of the following year (but don't rely on this).

3. Somewhere it says that cron sets the SHELL environment variable to /bin/sh, and elsewhere it implies that the default behaviour is for the user's default shell to be used to execute commands. Mcron sets the variable and runs the command in the user's default shell, as advertised by the /etc/passwd file.

3.3.1 Paul Vixie's copyright

3.3.2 Crontab files

A `crontab` file contains instructions to the `cron` daemon of the general form: "run this command at this time on this date". Each user has their own crontab, and commands in any given crontab will be executed as the user who owns the crontab. Uucp and News will usually have their own crontabs, eliminating the need for explicitly running `su` as part of a cron command.

Blank lines and leading spaces and tabs are ignored. Lines whose first non-space character is a pound-sign (#) are comments, and are ignored. Note that comments are not allowed on the same line as cron commands, since they will be taken to be part of the command. Similarly, comments are not allowed on the same line as environment variable settings.

An active line in a crontab will be either an environment setting or a cron command. An environment setting is of the form,

```
name = value
```

where the spaces around the equal-sign (=) are optional, and any subsequent non-leading spaces in `value` will be part of the value assigned to `name`. The `value` string may be placed in quotes (single or double, but matching) to preserve leading or trailing blanks.

Several environment variables are set up automatically by the `cron` daemon. SHELL is set to /bin/sh, and LOGNAME and HOME are set from the /etc/passwd line of the crontab's owner. HOME and SHELL may be overridden by settings in the crontab; LOGNAME may not.

(Another note: the LOGNAME variable is sometimes called USER on BSD systems... on these systems, USER will be set also.)[2]

In addition to LOGNAME, HOME, and SHELL, `cron` will look at MAILTO if it has any reason to send mail as a result of running commands in "this" crontab. If MAILTO is defined (and non-empty), mail is sent to the user so named. If MAILTO is defined but empty (MAILTO=""), no mail will be sent. Otherwise mail is sent to the owner of the crontab. This option is useful if you decide on /bin/mail instead of /usr/lib/sendmail as your mailer when you install cron – /bin/mail doesn't do aliasing, and UUCP usually doesn't read its mail.

The format of a cron command is very much the V7 standard, with a number of upward-compatible extensions. Each line has five time and date fields, followed by a user name if this is the system crontab file, followed by a command. Commands are executed by `cron` when the minute, hour, and month of year fields match the current time, **and** when at least one of the two day fields (day of month, or day of week) match the current time (see "Note" below). `cron` examines cron entries once every minute. The time and date fields are:

Field	Allowed values
minute	0-59
hour	0-23
day of month	0-31
month	0-12 (or names, see below)
day of week	0-7 (0 or 7 is Sun, or use names)

A field may be an asterisk (*), which always stands for "first-last".

Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an "hours" entry specifies execution at hours 8, 9, 10 and 11.

Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: "1,2,5,9", "0-4,8-12".

Step values can be used in conjunction with ranges. Following a range with "/<number>" specifies skips of the number's value through the range. For example, "0-23/2" can be used in the hours field to specify command execution every other hour (the alternative in the V7 standard is "0,2,4,6,8,10,12,14,16,18,20,22"). Steps are also permitted after an asterisk, so if you want to say "every two hours", just use "*/2".

[2] mcron has not been ported to BSD, so these notes are not relevant.

Names can also be used for the "month" and "day of week" fields. Use the first three letters of the particular day or month (case doesn't matter). Ranges or lists of names are not allowed.[3]

The "sixth" field (the rest of the line) specifies the command to be run. The entire command portion of the line, up to a newline or % character, will be executed by /bin/sh or by the shell specified in the SHELL variable of the cronfile. Percent-signs (%) in the command, unless escaped with backslash (\\), will be changed into newline characters, and all data after the first % will be sent to the command as standard input.

Note: The day of a command's execution can be specified by two fields – day of month, and day of week. If both fields are restricted (ie, aren't *), the command will be run when *either* field matches the current time. For example,

"30 4 1,15 * 5"

would cause a command to be run at 4:30 am on the 1st and 15th of each month, plus every Friday.

EXAMPLE CRON FILE

```
# use /bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to ‘paul’, no matter whose crontab this is
MAILTO=paul
#
# run five minutes after midnight, every day
5 0 * * *       $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 2:15pm on the first of every month -- output mailed to paul
15 14 1 * *     $HOME/bin/monthly
# run at 10 pm on weekdays, annoy Joe
0 22 * * 1-5 mail -s "It’s 10pm" joe%Joe,%%Where are your kids?%
23 0-23/2 * * * echo "run 23 minutes after midn, 2am, 4am ..., everyday"
5 4 * * sun     echo "run at 5 after 4 every sunday"
```

3.3.3 Extensions and incompatibilities

This section lists differences between Paul Vixie's cron and the olde-worlde BSD and AT&T programs, for the benefit of system administrators and users who are upgrading all the way.

- When specifying day of week, both day 0 and day 7 will be considered Sunday. BSD and AT&T seem to disagree about this.
- Lists and ranges are allowed to co-exist in the same field. "1-3,7-9" would be rejected by AT&T or BSD cron – they want to see "1-3" or "7,8,9" ONLY.
- Ranges can include "steps", so "1-9/2" is the same as "1,3,5,7,9".
- Names of months or days of the week can be specified by name.
- Environment variables can be set in the crontab. In BSD or AT&T, the environment handed to child processes is basically the one from /etc/rc.
- Command output is mailed to the crontab owner (BSD can't do this), can be mailed to a person other than the crontab owner (SysV can't do this), or the feature can be turned off and no mail will be sent at all (SysV can't do this either).

[3] Mcron allows any alphabetic characters after a name, so full names of days or months are also valid.

4 Detailed invoking

The program adopts one of three different personalities depending on the name used to invoke it. In a standard installation, the program is installed in the system under the names mcron, cron and crontab (installed SUID).

The recommended way to invoke the program is via the mcron personality described in the next section. The program can also be run as cron by root, and by the SUID program crontab by individual users to gain backwards compatibility with Vixie cron. However, due to the fact that this daemon process is shared by, and under control of, all the users of the system it is possible (though very unlikely) that it may become unusable, hence the recommendation to use the mcron personality.

Furthermore, the Vixie personality is considered deprecated by this author (it offers not a single advantage over the mcron personality, and bloats the code by a factor of three). It is unlikely that this personality will ever actually go away, but the program may in future be split into two distinct parts, and new developments will only take place in the part which implements the mcron personality.

4.1 Invoking mcron

Mcron should be run by the user who wants to schedule their jobs. It may be made a background job using the facilities of the shell. The basic command is `mcron [OPTION ...] [file ...]` which has the effect of reading all the configuration files specified (subject to the options) and then waiting until it is time to execute some command. If no files are given on the command line, then mcron will look in the user's cron configuration directories: these are ~/.cron (deprecated), the directory indicated by the `XDG_CONFIG_HOME` environment variable, or ~/.config/cron if this variable is not set. In any case, files which end in the extension .vixie or .vix will be assumed to contain Vixie-style crontabs, and files ending .guile or .gle will be assumed to contain Scheme code and will be executed as such; ANY OTHER FILES WILL BE IGNORED - specify a file name of "-" and then pipe the files into the standard input if you really want to read them, possibly using the `stdin` option to specify the type of file.

The program accepts the following options.

`-s count`
`--schedule=count`
: With this option specified no commands are run. Instead, the program computes the times the commands would be run and prints the information to the screen, and then immediately exits.

  The count indicates the number of commands to display.

`-d`
`--daemon`
: With this option the program will detach itself from the controlling terminal and run as a daemon process.

`-i (vixie|guile)`
`--stdin=(vixie|guile)`
: This option is used to indicate whether the configuration information being passed on the standard input is in Vixie format or Guile format. Guile is the default.

`--log-format`
: This option accepts an `(ice-9 format)` format string that can be used to customize the appearance of the output. The format string is applied to `format` with the following four arguments:

  1. A date/time string.
  2. The job process PID (as as number).
  3. The action name.
  4. The message to log.

  It defaults to `"~a ~2
  ~a: ~a~%"`, which produces output messages like:

  ```
  2021-08-17T12:01:01 some-job: completed in 0.218s
  ```

  If you'd rather see the job process PID instead of a timestamp, you could instead specify the format string as `"~1
  ~a ~a: ~a~%"`, which would result in something like:

  ```
  39234 some-job: completed in 0.218s
  ```

  To learn about all the possibilities offered by `(ice-9 format)`, refer to Section "Formatted Output" in *GNU Guile Reference Manual*.

`--date-format`
: This option accepts a `(srfi srfi-19)` date string format, to customize the appearance of the timestamp in output messages. It defaults to `"~5"`, which corresponds to a local ISO-8601 date/time format (see Section "SRFI-19 Date to string" in *GNU Guile Reference Manual*).

`-v`
`--version`
: This option causes a message to be printed on the standard output with information about the version and copyright for the current program.

`-h`
`--help`
: This causes a short but complete usage message to be displayed on standard output.

4.2 Invoking cron or crond

NOTE THAT THIS SECTION ONLY APPLIES IF THE `cron` or `crond`, and `crontab` PROGRAMS HAVE BEEN INSTALLED BY THE SYSTEM ADMINISTRATOR.

If the program runs by the name of `cron` or `crond`, then it will read all the files in `/var/cron/tabs` (which should only be readable by root) and the file `/etc/crontab`, and then detaches itself from the terminal to live forever as a daemon process. Additionally, it creates a UNIX socket at `/var/cron/socket`, and listens for messages sent to that socket consisting of a user name whose crontabs have been changed. In this case, the program will re-read that user's crontab. This is for correct functioning with the crontab program.

Further, unless the `--noetc` option is used, a job is scheduled to run every minute to check if `/etc/crontab` has been modified. If so, this file will also be re-read.

The options which may be used with this program are as follows.

`-v`
`--version`
: This option causes a message to be printed on the standard output with information about the version and copyright for the current program.

`-h`
`--help`
: This causes a short but complete usage message to be displayed on standard output.

`-s [count]`
`--schedule[=count]`
: With this option specified no commands are run. Instead, the program computes the times the commands would be run and prints the information to the screen, and then immediately exits.

  The count, if supplied, indicates the number of commands to display. The default value is 8.

`-n`
`--noetc`
: This tells cron not to add a job to the system which wakes up every minute to check for modifications to `/etc/crontab`. It is recommended that this option be used (and further that the `/etc/crontab` file be taken off the system altogether!)

`--log-format`
: Analogous to mcron's [–log-format], page 9.

`--date-format`
: Analogous to mcron's [–date-format], page 10.

4.3 Invoking crontab

This program is run by individual users to inspect or modify their crontab files. If a change is made to the file, then the root daemon process will be given a kick, and will immediately read the new configuration. A warning will be issued to standard output if it appears that a cron daemon is not running.

The command is used as

```
crontab [-u user] file
```

or

```
crontab [-u user] ( -l | -e | -r )
```

Only the root user can use the -u option, to specify the manipulation of another user's crontab file. In the first instance, the entire crontab file of the user is replaced with the contents of the specified file, or standard input if the file is "-".

In the latter case, the program behaves according to which of the (mutually exclusive) options was given (note that the long options are an mcron extension).

`-l`
`--list`
: Print the user's crontab file to the standard output, and exit.

`-r`

`--remove` Delete the user's crontab file, and exit.

`-e`

`--edit` Using the editor specified in the user's `VISUAL` or `EDITOR` environment variables, allow the user to edit their crontab. Once the user exits the editor, the crontab is checked for parseability, and if it is okay then it is installed as the user's new crontab and the daemon is notified that a change has taken place, so that the new file will become immediately effective.

4.4 Behaviour on laptops

While mcron has not been designed to work anachronistically, the behaviour of mcron when a laptop emerges from a suspended state is well defined, and the following description explains what happens in this situation.

When a laptop awakes from a suspended state, all jobs which would have run while the laptop was suspended will run exactly once immediately (and simultaneously) when the laptop awakes, and then the next time that those jobs run will be computed based on the time the laptop was awoken. Any jobs which would not have run during the suspense period will be unaffected, and will still run at their proper times.

4.5 Exit codes

The following are the status codes returned to the operating system when the program terminates.

0 No problems.

1 An attempt has been made to start cron but there is already a /var/run/cron.pid file. If there really is no other cron daemon running (this does not include invokations of mcron) then you should remove this file before attempting to run cron.

2 In parsing a guile configuration file, a `job` command has been seen but the second argument is neither a procedure, list or string. This argument is the job's action, and needs to be specified in one of these forms.

3 In parsing a guile configuration file, a `job` command has been seen but the first argument is neither a procedure, list or string. This argument is the job's next-time specification, and needs to be specified in one of these forms.

4 An attempt to run cron has been made by a user who does not have permission to access the crontabs in /var/cron/tabs. These files should be readable only by root, and the cron daemon must be run as root.

5 An attempt to run mcron has been made, but there are no jobs to schedule!

6 The system administrator has blocked this user from using crontab with the files /var/cron/allow and /var/cron/deny.

7 Crontab has been run with more than one of the arguments `-l`, `-r`, `-e`. These are mutually exclusive options.

8 Crontab has been run with the -u option by a user other than root. Only root is allowed to use this option.

9 An invalid Vixie-style time specification has been supplied.

10 An invalid Vixie-style job specification has been supplied.

11 A bad line has been seen in /etc/crontab.

12 The last component of the name of the program was not one of `mcron`, `cron`, `crond` or `crontab`.

13 Either none of the user's configuration directories exist, or there is a problem reading the files there. The configuration directories are ~/.cron and the directory pointed to by the `XDG_CONFIG_HOME` environment variable, or ~/.config/cron if this is not set.

15 Crontab has been run without any arguments at all. There is no default behaviour in this case.

16 Cron has been run by a user other than root.

5 Guile modules

Some of the key parts of mcron are implemented as modules so they can be incorporated into other Guile programs, or even into C-sourced programs if they are linked against libguile.

It may be, for example, that a program needs to perform house-keeping functions at certain times of the day, in which case it can spawn (either fork or thread) a sub-process which uses a built-in mcron. Another example may be a program which must sleep until some non-absolute time specified on the Gregorian calendar (the first day of next week, for example). Finally, it may be the wish of the user to provide a program with the functionality of mcron plus a bit extra.

The base module maintains mcron's internal job lists, and provides the main wait-run-wait loop that is mcron's primary function. It also introduces the facilities for accumulating a set of environment modifiers, which take effect when jobs run.

5.1 The base module

This module may be used by including `(use-modules (mcron base))` in a program. The main functions are `add-job` and `run-job-loop`, which allow a program to create a list of job specifications to run, and then to initiate the wait-run-wait loop firing the jobs off at the requisite times. However, before they are introduced two functions which manipulate the environment that takes effect when a job runs are defined.

The environment is a set of name-value pairs which is built up incrementally. Each time the `add-job` function is called, the environment modifiers that have been accumulated up to that point are stored with the new job specification, and when the job actually runs these name-value pairs are used to modify the run-time environment in effect.

`append-environment-mods` *name value* [Scheme procedure]
: When a job is run make sure the environment variable *name* has the value *value*.

`clear-environment-mods` [Scheme procedure]
: This procedure causes all the environment modifiers that have been specified so far to be forgotten.

`add-job` *time-proc action displayable configuration-time configuration-user* [*#:schedule* `%global-schedule`] [Scheme procedure]
: This procedure adds a job specification to the list of all jobs to run. *time-proc* should be a procedure taking exactly one argument which will be a UNIX time. This procedure must compute the next time that the job should run, and return the result. *action* should be a procedure taking no arguments, and contains the instructions that actually get executed whenever the job is scheduled to run. *displayable* should be a string, and is only for the use of humans; it can be anything which identifies or simply gives a clue as to the purpose or function of this job. *configuration-time* is the time from which the first invokation of this job should be computed. Finally, *configuration-user* should be the passwd entry for the user under whose personality the job is to run.

`run-job-loop` *fd-list* [*#:schedule* `%global-schedule`] [Scheme procedure]
: This procedure returns only under exceptional circumstances, but usually loops forever waiting for the next time to arrive when a job needs to run, running that job,

recomputing the next run time, and then waiting again. However, the wait can be interrupted by data becoming available for reading on one of the file descriptors in the fd-list, if supplied. Only in this case will the procedure return to the calling program, which may then make modifications to the job list before calling the `run-job-loop` procedure again to resume execution of the mcron base.

`remove-user-jobs` *user* [*#:schedule %global-schedule*] [Scheme procedure]
The argument *user* should be a string naming a user (their login name), or an integer UID, or an object representing the user's passwd entry. All jobs on the current job list that are scheduled to be run under this personality are removed from the job list.

`display-schedule` *count* [*port*] [*#:schedule %global-schedule*] [Scheme procedure]
This procedure is used to display a textual list of the next COUNT jobs to run.

The argument *count* must be an integer value giving the number of time-points in the future to report that jobs will run as. Note that this procedure is disruptive; if `run-job-loop` is called after this procedure, the first job to run will be the one after the last job that was reported in the schedule report. The report itself is returned to the calling program as a string.

`%date-format` [User Option]
This parameter holds the `(srfi srfi-19)` format string used to produce the time-stamp found in output messages. It defaults to `"~5"`.

`validate-date-format` *fmt* [Scheme procedure]
This procedure is used to validate *fmt*, a `(srfi srfi-19)` format string. When *fmt* is invalid, an error message is displayed and the program is aborted.

`%log-format` [User Option]
This parameter holds the `(ice-9 format)` format string used to produce the output messages. The four arguments applied to format are the timestamp, the process PID, the job name and the message. It defaults to `"~a ~2
~a: ~a~%"`.

`validate-log-format` *fmt* [Scheme procedure]
This procedure is used to validate *fmt*, a `(ice-9 format)` format string. When *fmt* is invalid, an error message is displayed and the program is aborted.

5.2 The redirect module

This module is introduced to a program with the command `(use-modules (mcron redirect))`.

This module provides the `with-mail-out` function, described fully in Section 3.1 [Guile Syntax], page 3.

5.3 The vixie-time module

This module is introduced to a program by `(use-modules (mcron vixie-time))`.

This module provides a single method for converting a Vixie-style time specification into a procedure which can be used as the `next-time-function` to the base `add-job` procedure, or to the `job-specifier job` procedure. See Section 3.3 [Vixie Syntax], page 6, for full details of the allowed format for the time string.

`parse-vixie-time` *time-string* [Scheme procedure]
: The single argument *time-string* should be a string containing a Vixie-style time specification, and the return value is the required procedure.

5.4 The job-specifier module

This module is introduced to a program by `(use-modules (mcron job-specifier))`.

This module provides all the functions available to user's Guile configuration files, namely `range`, `next-year-from`, `next-year`, `next-month-from`, `next-month`, `next-day-from`, `next-day`, `next-hour-from`, `next-hour`, `next-minute-from`, `next-minute`, `next-second-from`, `next-second`, and last but not least, `job`. See Section 3.1 [Guile Syntax], page 3, for full details.

Once this module is loaded, a Scheme configuration file can be used to put jobs onto the job list simply by `load`ing the file.

5.5 The vixie-specification module

To use this module, put the command `(use-modules (mcron vixie-specification))` into your program.

This module exports a couple of functions for adding jobs to the internal job list according to a Vixie-style crontab file.

`read-vixie-port` *port . parse-line* [Scheme procedure]
: This procedure reads a crontab from the given port, and adds jobs to the job list accordingly, taking care of environment specifications and comments which may appear in such a file.

  parse-line should not normally be used, except that if you are parsing a (deprecated) `/etc/crontab` file with a slightly modified syntax, you may pass the value *parse-system-vixie-line* as the optional argument.

`read-vixie-file` *name . parse-line* [Scheme procedure]
: This procedure attempts to open the named file, and if it fails will return silently. Otherwise, the behaviour is identical to `read-vixie-port` above.

Once this module has been declared in a program, a crontab file can be used to augment the current job list with a call to `read-vixie-file`.

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Z