CALCURSE - text-based organizer

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Introduction **1**

2 Overview **1**

2.1 Creation history . . . 1

2.2 Important features . . . 1

3 Installation **1**

3.1 Requirements . . . 1

3.1.1 ncurses library . . . 1

3.1.2 gettext library . . . 2

3.2 Install process . . . 2

4 calcurse basics **2**

4.1 Invocation . . . 2

4.1.1 Command line arguments . . . 2

4.1.2 Format strings . . . 4

Format specifiers for appointments . . . 4

Format specifiers for events . . . 5

Format specifiers for todo items . . . 5

Examples . . . 5

Extended format specifiers . . . 5

4.1.3 Environment variable for i18n . . . 6

4.1.4 Other environment variables . . . 6

4.2 User interface . . . 7

4.2.1 Non-interactive mode . . . 7

4.2.2 Interactive mode . . . 7

4.3 Background mode . . . 9

4.4 calcurse files . . . 9

4.5 Import/Export capabilities . . . 10

4.5.1 Import . . . 10

4.5.2 Export . . . 10

4.6 Online help . . . 11

5 Options **11**

5.1 General options . . . 11

5.2 Key bindings . . . 12

5.3 Color themes . . . 12

5.4 Layout configuration . . . 13

5.5 Sidebar configuration . . . 13

5.6 Notify-bar settings . . . 13

6 Known bugs **14**

7 Reporting bugs and feedback **14**

8 How to contribute? **14**

8.1 Translating documentation 14

8.2 calcurse i18n 15

8.2.1 Overview 15

8.2.2 Translator tasks 15

8.2.3 po-files 15

8.2.4 Uploading to Transifex 17

8.2.5 Using transifex-client 17

8.2.6 Using the Transifex web UI 17

9 Links **17**

9.1 calcurse homepage 17

9.2 calcurse announce list 18

9.3 Other links 18

10 Thanks **18**

Abstract

This manual describes `calcurse` functionalities, and how to use them. The installation from source is first described, together with the available command line arguments. The user interface is then presented, with all of the customizable options that change `calcurse` behavior. Last, bug reporting procedure is explained, as well as the way one can contribute to `calcurse` development.

1 Introduction

calcurse is a text-based calendar and scheduling application. It helps keeping track of events, appointments and everyday tasks. A configurable notification system reminds user of upcoming deadlines, and the curses based interface can be customized to suit user needs. All of the commands are documented within an online help system.

2 Overview

2.1 Creation history

Frederic started thinking about this project when he was finishing his Ph.D. in Astrophysics as it started to be a little hard to organize himself and he really needed a good tool for managing his appointments and todo list. Unfortunately, he finished his Ph.D. before finishing calcurse but he continued working on it, hoping it would be helpful to other people.

In mid-2010, Lukas took over development of calcurse and is now the main contributor and reviewer of patches.

But why calcurse anyway? Well, it is simply the concatenation of *cal*endar and *curse*s, the name of the library used to build the user interface.

2.2 Important features

Calcurse is multi-platform and intended to be lightweight, fast and reliable. It is to be used inside a console or terminal, locally or on a distant machine within an ssh (or similar) connection.

Calcurse can be run in two different modes : interactive or non-interactive mode. The first mode allows oneself to view its own personal organizer almost everywhere, thanks to the text-based interface. The second mode permits to easily build reminders just by adding calcurse with appropriate command line arguments inside a cron tab or within a shell init script.

Moreover, calcurse was created with the end-user in mind, and tends to be as friendly as possible. This means a complete on-line help system, together with having all of the possible actions displayed at any time inside a status bar. The user interface is configurable, and one can choose between several color and layout combinations. Key bindings are also configurable, to fit everyone's needs. Last, a configurable notification system reminds user of upcoming appointments. The reminders are sent even if the user's interface is not running, as calcurse is able to run in background.

3 Installation

3.1 Requirements

3.1.1 ncurses library

Calcurse requires only a C compiler, such as cc or gcc, and the ncurses library. It would be very surprising not to have a valid ncurses library already installed on your computer, but if not, you can find it at the following url: http://ftp.gnu.org/-pub/gnu/ncurses/

Note
It is also possible to link calcurse against the ncursesw library (ncurses with support for unicode).

3.1.2 gettext library

`calcurse` supports internationalization (**i18n** hereafter) through the `gettext` utilities. This means `calcurse` can produce multi-lingual messages if compiled with native language support (i.e. **NLS**).

However, **NLS** is optionnal and if you do not want to have support for multi-lingual messages, you can disable this feature. This is done by giving the `--disable-nls` option to `configure` (see section Install process). To check if the `gettext` utilities are installed on your system, you can search for the `libintl.h` header file for instance:

```
$ locate libintl.h
```

If this header file is not found, then you can obtain the `gettext` sources at the following url : http://ftp.gnu.org/pub/gnu/gettext/

Note
Even if `libintl.h` is found on your system, it can be wise to specify its location during the install process, by using the `--with-libintl-prefix` option with `configure`. Indeed, the `configure` could fail to locate this library if installed in an uncommon place.

3.2 Install process

First you need to gunzip and untar the source archive:

```
$ tar zxvf calcurse-3.2.1.tar.gz
```

Once you meet the requirements and have extracted the archive, the install process is quite simple, and follows the standard three steps process:

```
$ ./configure
$ make
$ make install     # (may require root privilege)
```

Use `./configure --help` to obtain a list of possible options.

4 calcurse basics

4.1 Invocation

4.1.1 Command line arguments

`calcurse` takes the following options from the command line (both short and long options are supported):

`-a, --appointment`
Print the appointments and events for the current day and exit. The calendar from which to read the appointments can be specified using the `-c` flag.

`-c <file>, --calendar <file>`
Specify the calendar file to use. The default calendar is `~/.calcurse/apts` (see section calcurse files). This option has precedence over `-D`.

`-d <date|num>, --day <date|num>`
Print the appointments for the given date or for the given number of upcoming days, depending on the argument format. Two possible formats are supported:

- a date (possible formats described below).
- a number `n`.

In the first case, the appointment list for the specified date will be returned, while in the second case the appointment list for the `n` upcoming days will be returned. As an example, typing `calcurse -d 3` will display your appointments for today, tomorrow, and the day after tomorrow. Possible formats for specifying the date are defined inside the general configuration menu (see General options), using the `format.inputdate` variable.

Note: as for the `-a` flag, the calendar from which to read the appointments can be specified using the `-c` flag.

`-D <dir>, --directory <dir>`
Specify the data directory to use. If not specified, the default directory is `~/.calcurse/`.

`--format-apt <format>`
Specify a format to control the output of appointments in non-interactive mode. See the Format strings section for detailed information on format strings.

`--format-recur-apt <format>`
Specify a format to control the output of recurrent appointments in non-interactive mode. See the Format strings section for detailed information on format strings.

`--format-event <format>`
Specify a format to control the output of events in non-interactive mode. See the Format strings section for detailed information on format strings.

`--format-recur-event <format>`
Specify a format to control the output of recurrent events in non-interactive mode. See the Format strings section for detailed information on format strings.

`--format-todo <format>`
Specify a format to control the output of todo items in non-interactive mode. See the Format strings section for detailed information on format strings.

`-g, --gc`
Run the garbage collector for note files and exit.

`-h, --help`
Print a short help text describing the supported command-line options, and exit.

`-i <file>, --import <file>`
Import the icalendar data contained in `file`.

`-l <num>, --limit <num>`
Limit the number of results printed to *num*.

`-n, --next`
Print the next appointment within upcoming 24 hours and exit. The indicated time is the number of hours and minutes left before this appointment.

Note: the calendar from which to read the appointments can be specified using the `-c` flag.

`-r[num], --range[=num]`
Print events and appointments for the num number of days and exit. If no num is given, a range of 1 day is considered.

`--read-only`
Don't save configuration nor appointments/todos.

Warning
Use this this with care! If you run an interactive calcurse instance in read-only mode, all changes from this session will be lost without warning!

`-s[date], --startday[=date]`
Print events and appointments from date and exit. If no date is given, the current day is considered.

-S<regex>, --search=<regex>
: When used with the `-a`, `-d`, `-r`, `-s`, or `-t` flag, print only the items having a description that matches the given regular expression.

--status
: Display the status of running instances of calcurse. If calcurse is running, this will tell if the interactive mode was launched or if calcurse is running in background. The process pid will also be indicated.

-t[num], --todo[=num]
: Print the `todo` list and exit. If the optional number `num` is given, then only todos having a priority equal to `num` will be returned. The priority number must be between 1 (highest) and 9 (lowest). It is also possible to specify `0` for the priority, in which case only completed tasks will be shown.

-v, --version
: Display `calcurse` version and exit.

-x[format], --export[=format]
: Export user data to specified format. Events, appointments and todos are converted and echoed to stdout. Two possible formats are available: ical and pcal (see section Links below). If the optional argument `format` is not given, ical format is selected by default.

  Note: redirect standard output to export data to a file, by issuing a command such as:

```
$ calcurse --export > my_data.dat
```

Note
The `-N` option has been removed in calcurse 3.0.0. See the Format strings section on how to print note along with appointments and events.

4.1.2 Format strings

Format strings are composed of printf()-style format specifiers — ordinary characters are copied to stdout without modification. Each specifier is introduced by a `%` and is followed by a character which specifies the field to print. The set of available fields depends on the item type.

Format specifiers for appointments

s
: Print the start time of the appointment as UNIX time stamp

S
: Print the start time of the appointment using the `hh:mm` format

d
: Print the duration of the appointment in seconds

e
: Print the end time of the appointment as UNIX time stamp

E
: Print the end time of the appointment using the `hh:mm` format

m
: Print the description of the item

n
: Print the name of the note file belonging to the item

N
: Print the note belonging to the item

Format specifiers for events

m
: Print the description of the item

n
: Print the name of the note file belonging to the item

N
: Print the note belonging to the item

Format specifiers for todo items

p
: Print the priority of the item

m
: Print the description of the item

n
: Print the name of the note file belonging to the item

N
: Print the note belonging to the item

Examples

`calcurse -r7 --format-apt='-%S -> %E\n\t%m\n%N'`
: Print appointments and events for the next seven days. Also, print the notes attached to each regular appointment (simulates `-N` for appointments).

`calcurse -r7 --format-apt=' -%m (%S to %E)\n' --format-recur-apt=' -%m (%S to %E)\n'`
: Print appointments and events for the next seven days and use a custom format for (recurrent) appointments: ` - Some appointment (18:30 to 21:30)`.

`calcurse -t --format-todo '(%p) %m\n'`
: List all todo items and put parentheses&#x200a;around the priority specifiers.

Extended format specifiers

Extended format specifiers can be used if you want to specify advanced formatting options. Extended specifiers are introduced by `%(` and are terminated by a closing parenthesis (`)`). The following list includes all short specifiers and corresponding long options:

- s: `(start)`
- S: `(start:epoch)`
- e: `(end)`
- E: `(end:epoch)`
- d: `(duration)`
- r: `(remaining)`
- m: `(message)`

- n: `(noteid)`
- N: `(note)`
- p: `(priority)`

The `(start)` and `(end)` specifiers support strftime()-style extended formatting options that can be used for fine-grained formatting. Additionally, the special formats `epoch` (which is equivalent to `(start:%s)` or `(end:%s)`) and `default` (which is mostly equivalent to `(start:%H:%M)` or `(end:%H:%M)` but displays `..:..` if the item doesn't start/end at the current day) are supported.

The `(remaining)` and `(duration)` specifiers support a subset of the strftime()-style formatting options, along with two extra qualifiers. The supported options are `%d`, `%H`, `%M` and `%S`, and by default each of these is zero-padded to two decimal places. To avoid the zero-padding, add `-` before the formatting option (for example, `%-d`). Additionally, the `E` option will display the total number of time units until the appointment, rather than showing the remaining number of time units modulo the next larger time unit. For example, an appointment in 50 hours will show as 02:00 with the formatting string `%H:%M`, but will show 50:00 with the formatting string `%EH:%M`. Note that if you are combining the `-` and `E` options, the `-` must come first. The default format for the `(remaining)` specifier is `%EH:%M`.

4.1.3 Environment variable for i18n

`calcurse` can be compiled with native language support (see gettext library). Thus, if you wish to have messages displayed into your native language, first make sure it is available by looking at the `po/LINGUAS` file. This file indicates the set of available languages by showing the two-letters corresponding code (for exemple, **fr** stands for french). If you do not find your language, it would be greatly appreciated if you could help translating `calcurse` (see the How to contribute? section).

If your language is available, run `calcurse` with the following command:

```
$ LC_ALL=fr_FR calcurse
```

i. where **fr_FR** is the locale name in this exemple, but should be replaced by the locale corresponding to the desired language.

You should also specify the charset to be used, because in some cases the accents and such are not displayed correctly. This charset is indicated at the beginning of the po file corresponding to the desired language. For instance, you can see in the fr.po file that it uses the iso-8859-1 charset, so you could run `calcurse` using the following command:

```
$ LC_ALL=fr_FR.ISO8859-1 calcurse
```

4.1.4 Other environment variables

The following environment variables affect the way `calcurse` operates:

VISUAL
: Specifies the external editor to use for writing notes.

EDITOR
: If the `VISUAL` environment variable is not set, then `EDITOR` will be used as the default external editor. If none of those variables are set, then `/usr/bin/vi` is used instead.

PAGER
: Specifies the default viewer to be used for reading notes. If this variable is not set, then `/usr/bin/less` is used.

4.2 User interface

4.2.1 Non-interactive mode

When called with at least one of the following arguments: `-a`, `-d`, `-h`, `-n`, `-t`, `-v`, `-x`, `calcurse` is started in non-interactive mode. This means the desired information will be displayed, and after that, `calcurse` simply quits and you are driven back to the shell prompt.

That way, one can add a line such as `calcurse --todo --appointment` in its init config file to display at logon the list of tasks and appointments scheduled for the current day.

4.2.2 Interactive mode

Note
Key bindings that are indicated in this manual correspond to the default ones, defined when `calcurse` is launched for the first time. If those key bindings do not suit user's needs, it is possible to change them within the keys configuration menu (see key bindings).

When called without any argument or only with the `-c` option, `calcurse` is started in interactive mode. In this mode, you are shown an interface containing three different panels which you can browse using the `TAB` key, plus a notification bar and a status bar (see figure below).

```
appointment panel---.                                   .---calendar panel
                    |                                   |
                    v                                   v
+------------------------------------++----------------------------+
|          Appointments              ||          Calendar          |
|------------------------------------||----------------------------|
|                 (|)  April 6, 2006 ||         April 2006         |
|                                    ||Mon Tue Wed Thu Fri Sat Sun |
|                                    ||                      1   2 |
|                                    ||  3   4   5   6   7   8   9 |
|                                    || 10  11  12  13  14  15  16 |
|                                    || 17  18  19  20  21  22  23 |
|                                    || 24  25  26  27  28  29  30 |
|                                    ||                            |
|                                    |+----------------------------+
|                                    |+----------------------------+
|                                    ||            ToDo            | todo
|                                    ||----------------------------| panel
|                                    ||                            |   |
|                                    ||                            |   |
|                                    ||                            |<--.
|                                    ||                            |
+------------------------------------++----------------------------+
|---[ Mon 2006-11-22 | 10:11:43 ]---(apts)----> 01:20 :: lunch <---|<--.
+------------------------------------------------------------------+ notify-bar
| ? Help     R Redraw    H/L -/+1 Day      G GoTo       C Config   |
| Q Quit     S Save      J/K -/+1 Week   Tab Chg View              |<-.
+------------------------------------------------------------------+  |
                                                                      |
                                                                status bar
```

The first panel represents a calendar which allows to highlight a particular day, the second one contains the list of the events and appointments on that day, and the last one contains a list of tasks to do but which are not assigned to any specific day.

Depending on the selected view, the calendar could either display a monthly (default as shown in previous figure) or weekly view. The weekly view would look like the following:

```
+------------------------------------+
|              Calendar              |
|----------------------------(# 13)--|
|    Mon Tue Wed Thu Fri Sat Sun     |
|     29  30  31  01  02  03  04     |
|                             <----+--  slice 1: 00:00 to 04:00 AM
|       --  --  --  --  --  --       |
|                             <----+--  slice 2: 04:00 to 08:00 AM
|       --  --  --  --  --  --       |
|                             <----+--  slice 3: 08:00 to 12:00 AM
|    -  --  --  --  --  --  --  -  <-+--  midday
|                             <----+--  slice 4: 12:00 to 04:00 PM
|       --  --  --  --  --  --       |
|                             <----+--  slice 5: 04:00 to 08:00 PM
|       --  --  --  --  --  --       |
|                             <----+--  slice 6: 08:00 to 12:00 PM
+------------------------------------+
```

The current week number is displayed on the top-right side of the panel (**# 13** meaning it is the 13th week of the year in the above example). The seven days of the current week are displayed in column. Each day is divided into slices of 4 hours each (6 slices in total, see figure above). A slice will appear in a different color if an appointment falls into the corresponding time-slot.

In the appointment panel, one can notice the `(|)` sign just in front of the date. This indicates the current phase of the moon. Depending on which is the current phase, the following signs can be seen:

` |) `
: first quarter

` (|) `
: full moon

` (| `
: last quarter

` | `
: new moon

no sign
: Phase of the moon does not correspond to any of the above ones.

At the very bottom of the screen there is a status bar, which indicates the possible actions and the corresponding keystrokes.

Just above this status bar is the notify-bar, which indicates from left to right : the current date, the current time, the calendar file currently in use (apts on the above example, which is the default calendar file, see the following section), and the next appointment within the upcoming 24 hours. Here it says that it will be lunch time in one hour and twenty minutes.

Note
Some actions, such as editing or adding an item, require to type in some text. This is done with the help of the built-in input line editor.

Within this editor, if a line is longer than the screen width, a >, `*`, or < character is displayed in the last column indicating that there are more character after, before and after, or before the current position, respectively. The line is scrolled horizontally as necessary.

Moreover, some editing commands are bound to particular control characters. Hereafter are indicated the available editing commands (^ stands for the control key):

`^a`
: moves the cursor to the beginning of the input line

^b
: moves the cursor backward

^d
: deletes one character forward

^e
: moves the cursor to the end of the input line

^f
: moves the cursor forward

^h
: deletes one character backward

^k
: deletes the input from the cursor to the end of the line

^w
: deletes backward to the beginning of the current word

ESCAPE
: cancels the editing

4.3 Background mode

When the daemon mode is enabled in the notification configuration menu (see Notify-bar settings), `calcurse` will stay in background when the user interface is not running. In background mode, `calcurse` checks for upcoming appointments and runs the user-defined notification command when necessary. When the user interface is started again, the daemon automatically stops.

‘calcurse` background activity can be logged (set the `daemon.log` variable in the notification configuration menu), and in that case, information about the daemon start and stop time, reminders’ command launch time, signals received… will be written in the `daemon.log` file (see section files).

Using the `--status` command line option (see section Command line arguments), one can know if `calcurse` is currently running in background or not. If the daemon is running, a message like the following one will be displayed (the pid of the daemon process will be shown):

```
calcurse is running in background (pid 14536)
```

Note
To stop the daemon, just send the `TERM` signal to it, using a command such as: `kill daemon_pid`, where **daemon_pid** is the process id of the daemon (14536 in the above example).

4.4 calcurse files

The following structure is created in your `$HOME` directory (or in the directory you specified with the -D option) the first time `calcurse` is run :

```
$HOME/.calcurse/
           |___notes/
           |___conf
           |___keys
           |___apts
           |___todo
```

`notes/`
: this subdirectory contains descriptions of the notes which are attached to appointments, events or todos. Since the file name of each note file is a SHA1 hash of the note itself, multiple items can share the same note file. calcurse provides a garbage collector (see the `-g` command line parameter) that can be used to remove note files which are no longer linked to any item.

`conf`
: this file contains the user configuration

`keys`
: this file contains the user-defined key bindings

`apts`
: this file contains all of the events and user's appointments

`todo`
: this file contains the todo list

Note
If the logging of calcurse daemon activity was set in the notification configuration menu, the extra file `daemon.log` will appear in calcurse data directory. This file contains logs about calcurse activity when running in background.

4.5 Import/Export capabilities

The import and export capabilities offered by `calcurse` are described below.

4.5.1 Import

Data in icalendar format as described in the rfc2445 specification (see links section below) can be imported into calcurse. Calcurse ical parser is based on version 2.0 of this specification, but for now on, only a subset of it is supported.

The following icalendar properties are handled by calcurse:

- `VTODO` items: "PRIORITY", "VALARM", "SUMMARY", "DESCRIPTION"
- `VEVENT` items: "DTSTART", "DTEND", "DURATION", "RRULE", "EXDATE", "VALARM", "SUMMARY", "DESCRIPTION"

The icalendar `DESCRIPTION` property will be converted into calcurse format by adding a note to the item. If a "VALARM" property is found, the item will be flagged as important and the user will get a notification (this is only applicable to appointments).

Here are the properties that are not implemented:

- negative time durations are not taken into account (item is skipped)
- some recurence frequences are not recognize: "SECONDLY" / "MINUTELY" / "HOURLY"
- some recurrence keywords are not recognized (all those starting with `BY`): "BYSECOND" / "BYMINUTE" / "BYHOUR" / "BYDAY" / "BYMONTHDAY" / "BYYEARDAY" / "BYWEEKNO" / "BYMONTH" / "BYSETPOS" plus "WKST"
- the recurrence exception keyword "EXRULE" is not recognized
- timezones are not taken into account

4.5.2 Export

Two possible export formats are available: `ical` and `pcal` (see section Links below to find out about those formats).

4.6 Online help

At any time, the built-in help system can be invoked by pressing the `?` key. By default, it shows an introduction to the help system in an external pager. You need to exit the pager in order to get back to calcurse (pressing `q` should almost always work). The default pager can be changed by setting the PAGER environment variable.

If you want to display help on a specific feature or key binding, type `:help <feature>` (e.g. `:help add`) or `:help <key>` (e.g. `:help ^A`) on the main screen.

5 Options

All of the `calcurse` parameters are configurable from the Configuration menu available when pressing `C`. You are then driven to a submenu with five possible choices : pressing `C` again will lead you to the Color scheme configuration, pressing `L` allows you to choose the layout of the main `calcurse` screen (in other words, where to put the three different panels on screen), pressing `G` permits you to choose between different general options, pressing `K` opens the key bindings configuration menu, and last you can modify the notify-bar settings by pressing `N`.

5.1 General options

These options control `calcurse` general behavior, as described below:

`general.autosave` (default: yes)
This option allows to automatically save the user's data (if set to **yes**) when quitting. <p class="rq"><span class="valorise">warnin
No data will be automatically saved if `general.autosave` is set to **no**. This means the user must press `S` (for saving) in order to retrieve its modifications.

`general.autogc` (default: no)
Automatically run the garbage collector for note files when quitting.

`general.periodicsave` (default: 0)
If different from `0`, user's data will be automatically saved every **general.periodicsave** minutes. When an automatic save is performed, two asterisks (i.e. `**`) will appear on the top right-hand side of the screen).

`general.confirmquit` (default: yes)
If set to **yes**, confirmation is required before quitting, otherwise pressing `Q` will cause `calcurse` to quit without prompting for user confirmation.

`general.confirmdelete` (default: yes)
If this option is set to **yes**, pressing `D` for deleting an item (either a **todo**, **appointment**, or **event**), will lead to a prompt asking for user confirmation before removing the selected item from the list. Otherwise, no confirmation will be needed before deleting the item.

`general.systemdialogs` (default: yes)
Setting this option to **no** will result in skipping the system dialogs related to the saving and loading of data. This can be useful to speed up the input/output processes.

`general.progressbar` (default: yes)
If set to **no**, this will cause the disappearing of the progress bar which is usually shown when saving data to file. If set to **yes**, this bar will be displayed, together with the name of the file being saved (see section calcurse files).

`appearance.calendarview` (default: 0)
If set to `0`, the monthly calendar view will be displayed by default otherwise it is the weekly view that will be displayed.

`general.firstdayofweek` (default: monday)
One can choose between Monday and Sunday as the first day of the week. If `general.firstdayofweek` is set to **monday**, Monday will be first in the calendar view. Otherwise, Sunday will be the first day of the week.

`format.outputdate` (default: `%D`)
This option indicates the format to be used when displaying dates in non-interactive mode. Using the default values, dates are displayed the following way: **mm/dd/aa**. You can see all of the possible formats by typing `man 3 strftime` inside a terminal.

`format.inputdate` (default: 1)
This option indicates the format that will be used to enter dates in **calcurse**. Four choices are available:

1. mm/dd/yyyy
2. dd/mm/yyyy
3. yyyy/mm/dd
4. yyyy-mm-dd

5.2 Key bindings

One can define ones own key bindings within the `Keys` configuration menu. The default keys look like the one used by the `vim` editor, especially the displacement keys. Anyway, within this configuration menu, users can redefine all of the keys available from within calcurse's user interface.

To define new key bindings, first highlight the action to which it will apply. Then, delete the actual key binding if necessary, and add a new one. You will then be asked to press the key corresponding to the new binding. It is possible to define more than one key binding for a single action.

An automatic check is performed to see if the new key binding is not already set for another action. In that case, you will be asked to choose a different one. Another check is done when exiting from this menu, to make sure all possible actions have a key associated with it.

The following keys can be used to define bindings:

- lower-case, upper-case letters and numbers, such as `a`, `Z`, `0`
- CONTROL-key followed by one of the above letters
- escape, horizontal tab, and space keys
- arrow keys (up, down, left, and right)
- `HOME` and `END` keys

While inside the key configuration menu, an online help is available for each one of the available actions. This help briefly describes what the highlighted action is used for.

Note
As of calcurse 3.0.0, displacement commands can be preceded by an optional number to repeat the command. For example, `10k` will move the cursor ten weeks forwards if you use default key bindings.

5.3 Color themes

`calcurse` color theme can be customized to suit user's needs. To change the default theme, the configuration page displays possible choices for foreground and background colors. Using arrows or calcurse displacement keys to move, and `X` or space to select a color, user can preview the theme which will be applied. It is possible to keep the terminal's default colors by selecting the corresponding choice in the list.

The chosen color theme will then be applied to the panel borders, to the titles, to the keystrokes, and to general informations displayed inside status bar. A black and white theme is also available, in order to support non-color terminals.

Note

Depending on your terminal type and on the value of the `$TERM` environment variable, color could or could not be supported. An error message will appear if you try to change colors whereas your terminal does not support this feature. If you do know your terminal supports colors but could not get `calcurse` to display them, try to set your `$TERM` variable to another value (such as **xterm-xfree86** for instance).

5.4 Layout configuration

The layout corresponds to the position of the panels inside `calcurse` screen. The default layout makes the calendar panel to be displayed on the top-right corner of the terminal, the todo panel on the bottom-right corner, while the appointment panel is displayed on the left hand-side of the screen (see the figure in section Interactive mode for an example of the default layout). By choosing another layout in the configuration screen, user can customize `calcurse` appearance to best suit his needs by placing the different panels where needed.

The following option is used to modify the layout configuration:

`appearance.layout` (default: 0)
Eight different layouts are to be chosen from (see layout configuration screen for the description of the available layouts).

5.5 Sidebar configuration

The sidebar is the part of the screen which contains two panels: the calendar and, depending on the chosen layout, either the todo list or the appointment list.

The following option is used to change the width of the sidebar:

`appearance.sidebarwidth` (default: 0)
Width (in percentage, 0 being the minimum width) of the side bar.

5.6 Notify-bar settings

The following options are used to modify the notify-bar behavior:

`appearance.notifybar` (default: yes)
This option indicates if you want the notify-bar to be displayed or not.

`format.notifydate` (default: %a %F)
With this option, you can specify the format to be used to display the current date inside the notification bar. You can see all of the possible formats by typing `man 3 strftime` inside a terminal.

`format.notifytime` (default: %T)
With this option, you can specify the format to be used to display the current time inside the notification bar. You can see all of the possible formats by typing `man 3 strftime` inside a terminal.

`notification.warning` (default: 300)
When there is an appointment which is flagged as `important` within the next `notification.warning` seconds, the display of that appointment inside the notify-bar starts to blink. Moreover, the command defined by the `notification.command` option will be launched. That way, the user is warned and knows there will be soon an upcoming appointment.

`notification.command` (default: printf \a)
This option indicates which command is to be launched when there is an upcoming appointment flagged as `important`. This command will be passed to the user's shell which will interpret it. To know what shell must be used, the content of the `$SHELL` environment variable is used. If this variable is not set, `/bin/sh` is used instead.

Say the `mail` command is available on the user's system, one can use the following command to get notified by mail of an upcoming appointment (the appointment description will also be mentioned in the mail body):

```
$ calcurse --next | mail -s "[calcurse] upcoming appointment!" user@host.com
```

`notification.notifyall` (default: no)
Invert the sense of flagging an appointment as `important`. If this is enabled, all appointments will be notified - except for flagged ones.

`daemon.enable` (default: no)
If set to yes, daemon mode will be enabled, meaning `calcurse` will run into background when the user's interface is exited. This will allow the notifications to be launched even when the interface is not running. More details can be found in section Background mode.

`daemon.log` (default: no)
If set to yes, `calcurse` daemon activity will be logged (see section files).

6 Known bugs

Incorrect highlighting of items appear when using calcurse black and white theme together with a `$TERM` variable set to **xterm-color**. To fix this bug, and as advised by Thomas E. Dickey (`xterm` maintainer), **xterm-xfree86** should be used instead of **xterm-color** to set the `$TERM` variable:

> "The xterm-color value for $TERM is a bad choice for XFree86 xterm because it is commonly used for a `terminfo` entry which happens to not support bce. Use the xterm-xfree86 entry which is distributed with XFree86 xterm (or the similar one distributed with ncurses)."

7 Reporting bugs and feedback

Please send bug reports and feedback to: `misc .at.calcurse .dot.org`.

8 How to contribute?

If you would like to contribute to the project, you can first send your feedback on what you like or dislike, and if there are features you miss in `calcurse`. For now on, possible contributions concern the translation of `calcurse` messages and documentation.

8.1 Translating documentation

Note
We recently dropped all translations of the manual files from the distribution tarball. There are plan to reintroduce them in form of a Wiki on the calcurse website. Please follow the mailing lists for up-to-date information.

The **doc/** directory of the source package already contains translated version of `calcurse` manual. However, if the manual is not yet available into your native language, it would be appreciated if you could help translating it.

To do so, just copy one of the existing manual file to `manual_XX.html`, where **XX** identifies your language. Then translate this newly created file and send it to the author (see Reporting bugs and feeback), so that it can be included in the next `calcurse` release.

8.2 calcurse i18n

As already mentioned, `gettext` utilities are used by `calcurse` to produce multi-lingual messages. We are currently using Transifex to manage those translations.

This section provides informations about how to translate those messages into your native language. However, this howto is deliberately incomplete, focusing on working with `gettext` for `calcurse` specifically. For more comprehensive informations or to grasp the Big Picture of Native Language Support, you should refer to the `GNU gettext` manual at: http://www.gnu.org/-software/gettext/manual/

Important
Since we switched to Transifex, editing **po** files is not necessary anymore as Transifex provides a user-friendly, intuitive web interface for translators. Knowledge of `gettext` and the **po** format is still useful for those of you who prefer the command line and local editing. If you want to use the web UI to edit the translation strings, you can skip over to Using the Transifex web UI tough.

Basically, three different people get involved in the translation chain: coders, language coordinator, and translators. After a quick overview of how things work, the translator tasks will be described hereafter.

8.2.1 Overview

To be able to display texts in the native language of the user, two steps are required: **internationalization** (i18n) and **localization** (l10n).

i18n is about making `calcurse` support multiple languages. It is performed by coders, who will mark translatable texts and provide a way to display them translated at runtime.

l10n is about making the i18n'ed `calcurse` adapt to the specific language of the user, ie translating the strings previously marked by the developers, and setting the environment correctly for `calcurse` to use the result of this translation.

So, translatable strings are first marked by the coders within the `C` source files, then gathered in a template file (**calcurse.pot** - the **pot** extension meaning **portable object template**). The content of this template file is then merged with the translation files for each language (**fr.po** for French, for instance - with **po** standing for **portable object**, ie meant to be read and edited by humans). A given translation team will take this file, translate its strings, and send it back to the developers. At compilation time, a binary version of this file (for efficiency reasons) will be produced (**fr.mo** - **mo** stands for **machine object**, i.e. meant to be read by programs), and then installed. Then `calcurse` will use this file at runtime, translating the strings according to the locale settings of the user.

8.2.2 Translator tasks

Suppose someone wants to initiate the translation of a new language. Here are the steps to follow:

- First, find out what the locale name is. For instance, for french, it is `fr_FR`, or simply `fr`. This is the value the user will have to put in his `LC_ALL` environment variable for software to be translated (see Environment variable for i18n).
- Then, go into the **po/** directory, and create a new po-file from the template file using the following command: `msginit -i calcurse.pot -o fr.po -l fr --no-translator` If you do not have `msginit` installed on your system, simply copy the **calcurse.pot** file to **fr.po** and edit the header by hand.

Now, having this **fr.po** file, the translator is ready to begin.

8.2.3 po-files

The format of the po-files is quite simple. Indeed, po-files are made of four things:

1. **location lines:** tells you where the strings can be seen (name of file and line number), in case you need to see a bit of context.
2. **msgid lines:** the strings to translate.
3. **msgstr lines:** the translated strings.
4. **lines prefixed with #:** comments (some with a special meaning, as we will see below).

Basically, all you have to do is fill the **msgstr** lines with the translation of the above **msgid** line.

A few notes:

Fuzzy strings
: You will meet strings marked with a `"#, fuzzy"` comment. `calcurse` won't use the translations of such strings until you do something about them. A string being fuzzy means either that the string has already been translated but has since been changed in the sources of the program, or that this is a new string for which `gettext` made a *wild guess* for the translation, based on other strings in the file. It means you have to review the translation. Sometimes, the original string has changed just because a typo has been fixed. In this case, you won't have to change anything. But sometimes, the translation will no longer be accurate and needs to be changed. Once you are done and happy with the translation, just remove the `"#, fuzzy"` line, and the translation will be used again in `calcurse`.

c-format strings and special sequences
: Some strings have the following comment: `"#, c-format"`. This tells that parts of the string to translate have a special meaning for the program, and that you should leave them alone. For instance, %-sequences, like `"%s"`. These means that `calcurse` will replace them with another string. So it is important it remains. There are also \-sequences, like `\n` or `\t`. Leave them, too. The former represents an end of line, the latter a tabulation.

Translations can be wrapped
: If lines are too long, you can just break them like this:

```
msgid ""
"some very long line"
"another line"
```

po-file header
: At the very beginning of the po-file, the first string form a header, where various kind of information has to be filled in. Most important one is the charset. It should resemble

```
"Content-Type: text/plain; charset=utf-8\n"
```

You should also fill in the Last-Translator field, so that potential contributors can contact you if they want to join you in the translation team, or have remarks/typo fixes to give about the translations. You can either just give your name/nick, or add an email address, for exemple:

```
"Last-Translator: Frederic Culot <frederic@culot.org>\n"
```

Comments
: Adding comments (lines begining with the # character) can be a good way to point out problems or translation difficulties to proofreaders or other members of your team.

Strings size
: `calcurse` is a curses/console program, thus it can be heavily dependant on the terminal size (number of columns). You should think about this when translating. Often, a string must fit into a single line (standard length is 80 characters). Don't translate blindly, try to look where your string will be displayed to adapt your translation.

A few useful tools
: The po-file format is very simple, and the file can be edited with a standard text editor. But if you prefer, there are few specialized tools you may find convenient for translating:

- `poEdit` (http://www.poedit.org/)

- `KBabel` (http://i18n.kde.org/tools/kbabel/)
- `GTranslator` (http://gtranslator.sourceforge.net/)
- `Emacs` po mode
- `Vim` po mode

And finally
I hope you'll have fun contributing to a more internationalized world. :) If you have any more questions, don't hesitate to contact us at **misc .at. calcurse .dot. org**.

8.2.4 Uploading to Transifex

There's several different ways to upload a finished (or semi-finished) translation for a new language to Transifex. The easiest way is to browse to the Translation Teams page and request the addition of a new team.

As soon as we accepted your request, you will be able to upload your **po** file on the calcurse resource page by clicking the **Add translation** button at the bottom.

8.2.5 Using transifex-client

You can also use a command line client to submit translations instead of having to use the web interface every time you want to submit an updated version. If you have a recent version of `setuptools` installed, you can get the CLI client by issuing the following command:

```
$ easy_install -U transifex-client
```

Alternatively, you can get the source code of transifex-client at http://pypi.python.org/pypi/transifex-client.

After you downloaded and installed the client, run the following commands in the calcurse source directory to checkout the translation resources of **calcurse**:

```
$ tx pull -a
```

To submit changes back to the server, use:

```
$ tx push -r calcurse.calcursepot -t -l <locale>
```

For more details, read up on `transifex-client` usage at the The Transifex Command-line Client v0.4 section of the Transifex documentation.

8.2.6 Using the Transifex web UI

As an alternative to editing **po** files, there is a web-based interface that can be used to create and update translations. After having signed up and created a new translation team (see Uploading to Transifex on how to do that), click the **Add translation** button at the bottom on the calcurse resource page, select the language you'd like to translate and choose **Translate Online**.

9 Links

This section contains links and references that may be of interest to you.

9.1 calcurse homepage

The `calcurse` homepage can be found at http://calcurse.org

9.2 calcurse announce list

If you are interested in the project and want to be warned when a new release comes out, you can subscribe to the `calcurse` announce list. In doing so, you will receive an email as soon as a new feature appears in `calcurse`.

To subscribe to this list, send a message to **announce+subscribe .at. calcurse .dot. org** with "subscribe" in the subject field.

9.3 Other links

You may want to look at the ical format specification (rfc2445) at: http://tools.ietf.org/html/rfc2445

The pcal project page can be found at: http://pcal.sourceforge.net/

10 Thanks

Its time now to thank other people without whom this program would not exist! So here is a list of contributing persons I would like to thank :

- Alex for its patches, help and advices with `C` programming
- Gwen for testing and general discussions about how to improve `calcurse`
- Herbert for packaging `calcurse` for FreeBSD
- Zul for packaging `calcurse` for NetBSD
- Wain, Steffen and Ronald for packaging `calcurse` for Archlinux
- Kevin, Ryan, and fEnIo for packaging `calcurse` for Debian and Ubuntu
- Pascal for packaging `calcurse` for Slackware
- Alexandre and Markus for packaging `calcurse` for Mac OsX and Darwin
- Igor for packaging `calcurse` for ALT Linux
- Joel for its calendar script which inspired `calcurse` calendar view
- Jeremy Roon for the Dutch translation
- Frédéric Culot, Toucouch, Erik Saule, Stéphane Aulery and Baptiste Jonglez for the French translation
- Michael Schulz, Chris M., Benjamin Moeller and Lukas Fleischer for the German translation
- Rafael Ferreira for the Portuguese (Brazil) translation
- Aleksey Mechonoshin for the Russian translation
- Jose Lopez for the Spanish translation
- Tony for its patch which helped improving the recur_item_inday() function, and for implementing the date format configuration options
- Erik Saule for its patch implementing the `-N`, `-s`, `-S`, `-r` and `-D` flags
- people who write softwares I like and which inspired me, especially :
  - `vim` for the displacement keys
  - `orpheus` and `abook` for documentation
  - `pine` and `aptitude` for the text user interface
  - `tmux` for coding style

And last, many many thanks to all of the `calcurse` users who sent me their feedback.