

A Survey of GNU Guile Software

Erik Edrosa

March 23, 2019

About me

- GNU Guile user since 2014
- Maintainer of Guile-CommonMark and Guile-Git
- C++ Software Developer from Miami, Florida

What is GNU Guile?



- GNU Ubiquitous Intelligent Language for Extensions
- The official extension language of the GNU project.
- Implementation of the Scheme programming language.

Origins of GNU Guile

- GNU Hackers were inspired by the customability and extendability of GNU Emacs
- They wanted to bring this to the rest of GNU
- Decided to use scheme, because it is simple and clean
- Should support multiple languages like Emacs Lisp and Tcl

Definitions

Customizable easily alter the behavior of the software.

Extensible can go beyond simple customizations and create new ways to use the software.

GNU Emacs

- An extensible, customizable, free/libre text editor
- Also a calendar, email client, package manager, web browser, ...
- and more!
- Find out more at <https://www.gnu.org/s/emacs>

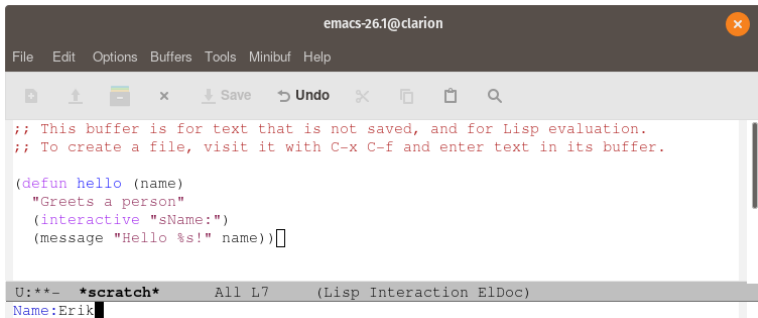


Extending Emacs with Commands

- Uses a command loop
- First reads a key sequence
- Key sequence is translated to a command
- The command is executed

```
(defun hello (name)
  "Greet a person"
  (interactive "sName:")
  (message "Hello %s!" name))
```

Example of an Emacs Command

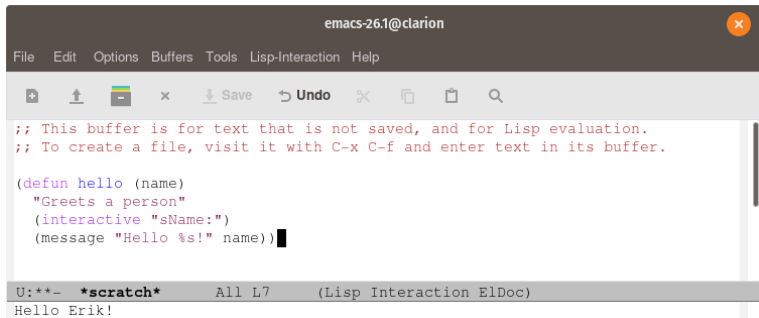


The screenshot shows the Emacs editor window titled "emacs-26.1@clarion". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Minibuf", and "Help". The toolbar contains icons for file operations and editing, along with "Save", "Undo", and search icons. The main text area contains the following Lisp code:

```
;; This buffer is for text that is not saved, and for Lisp evaluation.  
;; To create a file, visit it with C-x C-f and enter text in its buffer.  
  
(defun hello (name)  
  "Greet a person"  
  (interactive "sName:")  
  (message "Hello %s!" name))
```

The status bar at the bottom of the window shows "U:**- *scratch* All L7 (Lisp Interaction ElDoc)". The minibuffer at the bottom of the window shows "Name:Erik" with a cursor at the end.

Example of an Emacs Command



The screenshot shows the Emacs editor window titled "emacs-26.1@clarion". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Lisp-Interaction", and "Help". The toolbar contains icons for file operations and editing, along with "Save", "Undo", and search icons. The main text area contains the following code:

```
;; This buffer is for text that is not saved, and for Lisp evaluation.  
;; To create a file, visit it with C-x C-f and enter text in its buffer.  
  
(defun hello (name)  
  "Greet a person"  
  (interactive "sName:")  
  (message "Hello %s!" name))
```

The status bar at the bottom of the window displays "U:**- *scratch* All L7 (Lisp Interaction ElDoc)" and "Hello Erik!".

Customizing Emacs

- Customizable variables or "user options"
- Can do simple customizations with `customize` command
- Emacs will save these to a file for you
- Or users can add to their init file
- `(setq tab-width 4)`
- Emacs makes it easy for extensions to add their own with `defcustom`

Emacs is Self-Documenting

The screenshot shows the Emacs editor window titled 'emacs-26.1@clarion'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Emacs-Lisp', and 'Help'. The toolbar contains icons for file operations and search. The main text area shows the following code:

```
(setq tab-width 4)
```

A help window is displayed below the code, showing the documentation for the variable `tab-width` from the file `init.el`. The help text includes:

```

-:***-  init.el      All L2      (Emacs-Lisp ElDoc)
tab-width is a variable defined in 'C source code'.
Its value is 4
Original value was 8
Local in buffer init.el; global value is 8

  Automatically becomes buffer-local when set.
  This variable is safe as a file local variable if its value
  satisfies the predicate 'integerp'.

Documentation:
Distance between tab stops (for display of tab characters), in columns.
NOTE: This controls the display width of a TAB character, and not
the size of an indentation step.
This should be an integer greater than zero.

You can customize this variable.

U:%%-  *Help*      Top L1      (Help)

```

What is GNU Guix?



- GNU Guix is a package management tool
- Uses a declarative approach to define packages
- as well as configure operating systems.
- Find out more at <https://www.gnu.org/s/guix>

A Guix Package Definition

```
(define-public hello
  (package
    (name "hello")
    (version "2.10")
    (source (origin
              (method url-fetch)
              (uri (string-append "mirror://gnu/hello/hello-" version
                                   ".tar.gz"))
              (sha256
                (base32
                  "0ssi1wpaf7plawqqjwigppsg5fyh99vdlb9kzl7c9lmg89ndq1i")))))
    (build-system gnu-build-system)
    (synopsis "Hello, GNU world: An example GNU package")
    (description
      "GNU Hello prints the message \"Hello, world!\" and then exits. It
serves as an example of standard GNU coding practices. As such, it supports
command-line arguments, multiple languages, and so on.")
    (home-page "https://www.gnu.org/software/hello/")
    (license gpl3+)))
```

A Guix Package Using Inheritance

```
(define-public guile2.0-commonmark
  (package
    (inherit guile-commonmark)
    (name "guile2.0-commonmark")
    (inputs '(("guile" ,guile-2.0)))))
```

A Guix Operating System Configuration

```
(operating-system
  (host-name "antelope")
  (timezone "Europe/Paris")
  (locale "en_US.utf8")

  (bootloader (bootloader-configuration
    (bootloader grub-efi-bootloader)
    (target "/boot/efi")))

  (mapped-devices
    (list (mapped-device
      (source (uuid "12345678-1234-1234-1234-123456789abc"))
      (target "my-root")
      (type luks-device-mapping))))

  (file-systems (append
    (list (file-system
      (device (file-system-label "my-root"))
      (mount-point "/")
      (Type "ext4")
      (dependencies mapped-devices))
      %base-file-systems))

  (users (cons (user-account
    (name "bob")
    (comment "Alice's brother")
    (group "users")
    (supplementary-groups '("wheel" "netdev"
      "audio" "video")))
    (home-directory "/home/bob"))
    %base-user-accounts))

  (packages (append (list
    nss-certs
    gvfs)
    %base-packages))

  (services (append (list (gnome-desktop-service)
    (xfce-desktop-service))
    %desktop-services))

  (name-service-switch %mdns-host-lookup-nss))
```

Extending Guix with Scheme

```
(use-modules (guix packages)
             (gnu packages package-management))

(map car (package-inputs guix))
```

```
'("bzip2"
  "gzip"
  "zlib"
  "sqlite"
  "libgcrypt"
  "guile"
  "boot-guile"
  "util-linux"
  "boot-guile/i686")
```


What is GnuCash?

- A personal and small-business accounting software.
- Uses Double-Entry Accounting.
- Uses Guile to allow users to create custom reports.
- Find out more at <https://gnucash.org>



Creating a GnuCash Custom Report

```
(gnc:define-report
 'version 1
 'name (N_ "An Example")
 'report-guid "a unique id"
 'menu-name (N_ "Example Report")
 'menu-tip (N_ "Some example")
 'menu-path (list gnc:menuname-utility)
 'options-generator options-generator
 'renderer document-renderer)
```

- report version
- name of the report
- unique id
- name of the report in the menu
- tip to provide additional information
- path to place report in the menu
- procedure that generates report options
- procedure that generates the html document

GnuCash Option Generator

```
(define (options-generator)
  (let ((options (gnc:new-options)))
    (gnc:register-option
     options
     (gnc:make-simple-boolean-option
      (N_ "Section 1")
      (N_ "Option Name")
      "a"
      (N_ "Help text")
      #t))
    (gnc:options-set-default-section
     options "Section 1")
    options))
```

- Create an options object
- Register option object
- Create a boolean option
- Section to put the option
- Name of the option
- Sorting key
- Help tool tip text
- Default value
- Default section to display

GnuCash Document Renderer

```
(define (document-renderer report)
  (let* ((document (gnc:make-html-document))
        (report-options (gnc:report-options report))
        (option (gnc:lookup-option report-options
                                   "Section 1"
                                   "Option Name"))
        (value (gnc:option-value option)))
    (gnc:html-document-set-title! document (_ "Example Report"))
    (gnc:html-document-add-object!
     document
     (gnc:make-html-text
      (gnc:html-markup-p
       (if value
           (_ "Option was True!")
           (_ "Option was False!")))))
    document))
```

What is GNU Lilypond?



- A music engraving program.
- Use a text based notation for music input
- Similar idea to using \LaTeX
- Find out more at lilypond.org

Lilypond input language



```
\version "2.18.2"

foo = \relative c' {
  c d e f
}

{
  \foo
}
```

Lilypond internal representation

```
(make-music
 'RelativeOctaveMusic
 'element (make-music
 'SequentialMusic
 'elements (list (make-music
 'NoteEvent
 'pitch (ly:make-pitch 0 0 0)
 'duration (ly:make-duration 2 0 1))
 (make-music
 'NoteEvent
 'pitch (ly:make-pitch 0 1 0)
 'duration (ly:make-duration 2 0 1))
 (make-music
 'NoteEvent
 'pitch (ly:make-pitch 0 2 0)
 'duration (ly:make-duration 2 0 1))
 (make-music
 'NoteEvent
 'pitch (ly:make-pitch 0 3 0)
 'duration (ly:make-duration 2 0 1))))))
```

Using Scheme to extend Lilypond



```
foo = \relative c' {  
  c d e f  
}
```

```
#(define (twice x)  
  (make-sequential-music (list x x)))
```

```
#(define bar (twice foo))
```

```
\bar
```


Guile Supports Multiple Languages

- Wisp, indentation based syntax into s-expression
- Lua, using Guile-Lua
- Theme-D, a language that extends Scheme with static typing
- ECMAScript, also known as JavaScript
- Emacs Lisp, what GNU Emacs uses

Extending Software Using Commands

Can we bring the "editor command loop" from Emacs to other software?
Emacsy, the embeddable Emacs-like library for Guile.

Self-Documenting Software

Another powerful feature from Emacs.
How should help be displayed?

What else?

How else can we extend our software?

Join Us

Join the community on Freenode irc channel `#guile`
Learn more about GNU Guile at www.gnu.org/s/guile

Thank You

© 2019 Erik Edrosa

This work is licensed under Creative Commons Attribution Share-Alike 4.0 International

Attributions

"Logo proposal for GNU Guile" by Luis Felipe López Acevedo is licensed under CC BY-SA 4.0 / Crop from original

"GNU Guix Logotype" by Luis Felipe López Acevedo is licensed under CC BY-SA 4.0 / Crop from original

"Official LilyPond logo" by Han-Wen Nienhuys, Jan Nieuwenhuis is licensed under CC BY-SA 3.0