Minipaca II User Manual

Xu Xin

sxuxin@protonmail.com

Contents

1 Introduction

Minipaca is a tool for software development. It uses Chez Scheme as a programming language and GTK4 as a graphical interface library to develop software.
Minipaca has built-in interface bindings between Chez Scheme and GTK4, so whether you develop command-line software or graphical interface software, you can use Chez Scheme as a programming language and do not need to use C language.
Minipaca compiles the source files into an executable file, and then packages the executable and other files into the installation package for each distribution. Currently, Minipaca supports numerous distributions under the three platforms of Linux, BSD, and Windows.
Minipaca was developed using its own technology.

Minipaca:	`https://minipaca.github.io`
Chez Scheme:	`https://github.com/cisco/ChezScheme`
GTK4:	`https://docs.gtk.org/gtk4`

2 Installation

- X86 _64 / AMD64

	Distro	Abbr.	Format
Linux	Arch Linux	arch	pkg.tar.zst
Linux	EndeavourOS	endeavour	pkg.tar.zst
Linux	Debian Testing	debiant	deb
Linux	Debian 12	debian12	deb
Linux	Ubuntu 21.10	ub2110	deb
Linux	Ubuntu 22.04	ub2204	deb
Linux	Ubuntu 23.04	ub2304	deb
Linux	OpenSUSE Tumbleweed	opensusetw	rpm
Linux	Fedora Linux 36	fc36	rpm
Linux	Fedora Linux 37	fc37	rpm
Linux	Fedora Linux 38	fc38	rpm
Linux	Void Linux	void	xbps
BSD	FreeBSD 13	fb13	pkg
Windows	Windows 10	win10	exe
Windows	Windows 11	win11	exe

- AArch64

	Distro	Abbr.	Format
Linux	Arch Linux for AArch64	arch-aarch64	pkg.tar.zst

Please go to the Minipaca website and select the package closest to your system to download and follow the instructions below to install it:

- Arch Linux, EndeavourOS:

```
yay -S chez-scheme
sudo pacman -U ./xx.pkg.tar.zst
```

- Arch Linux (aarch64):

```
yay -S chez-scheme-racket-git
sudo pacman -U ./xx.pkg.tar.zst
```

- Ubuntu 22.04+, Debian 12+, Debian Testing:

```
sudo apt install ./xx.deb
```

- openSUSE Tumbleweed:

```
sudo zypper addreop https://download.opensuse.org/\
repositories/devel:languages:misc/\
openSUSE_Tumbleweed/devel:languages:misc.repo
sudo zypper refresh
sudo zypper install ./xx.rpm
```

- Fedora 36+:

```
sudo dnf copr enable superboum/chez-scheme
sudo dnf install ./xx.rmp
```

- Void Linux:

```
xbps-rindex -a xx.x86_64.xbps
sudo xbps-install -R ./ xx
```

- FreeBSD 13+:

```
pkg add -f xx.pkg
```

- Windows 10+:

Double-click to install

3 First time to use

Open a terminal software, type minipaca, and press Enter:

```
$ minipaca

    Build Cross Platform Apps With Chez Scheme And GTK4

Usage:

    minipaca -x [command]
    minipaca <shortcut>

The commands are:

    app         create a CLI project 'app'
    app-gtk4    create a GUI project 'app-gtk4'

    generate    generate an executable
    pack        pack for distros

    version     print version
    usage       show usage

    functree    show function call tree
    translate   translte automatically

    user        user infomation
    reference   find API reference

The shortcuts are:

    -g          shortcut for '-x generate'
    -p          shortcut for '-x pack'
```

4 Develop your first CLI software

For convenience, it is recommended to develop the software in a special folder, such as the *dev* folder.

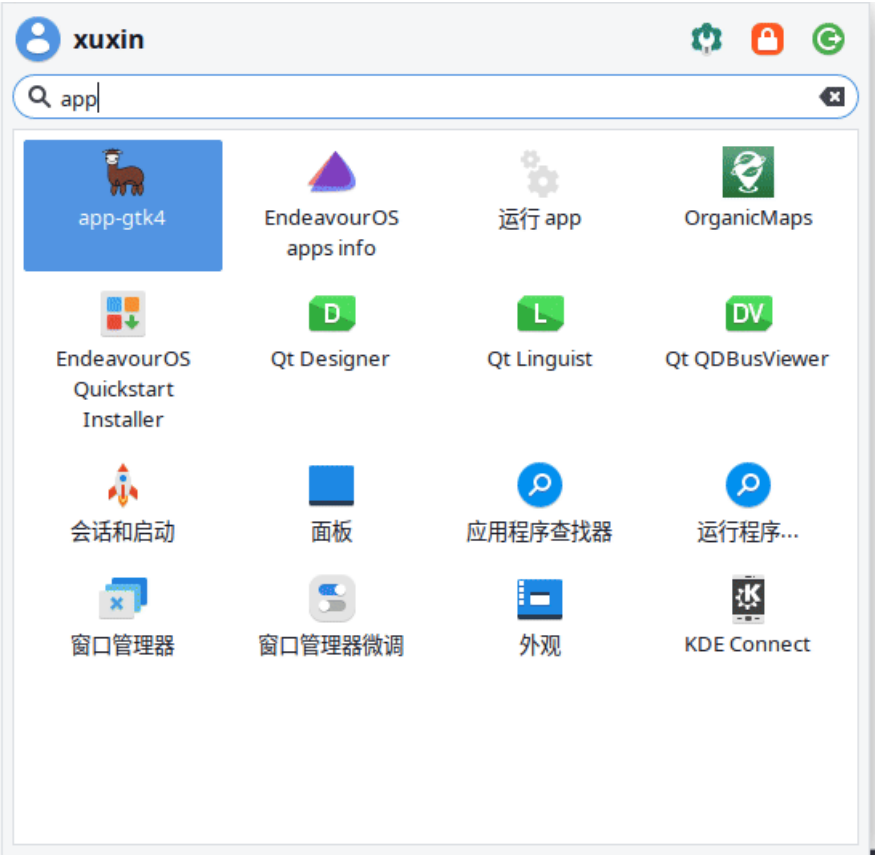
- Open a terminal software, type *mkdir dev*, and press Enter to create a new *dev* folder;
- Type *cd dev* and press Enter to enter the *dev* folder;
- Type *minipaca -x app* and press Enter to create a new software *app*;
- Type *cd app* and press Enter to enter the *app* folder;
- Type *minipaca -g* and press Enter to compile the *app*;
- Type *minipaca -p* and press Enter to package the *app*;
- Open the file manager and navigate to ~ → dev → app → p to view the packaged package;
- Install the *app* according to the same installation method mentioned earlier;
- Type *app* and press Enter to run the *app*;

Congratulations, you have successfully developed a command line software.

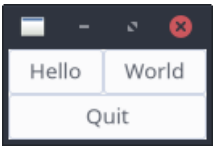
5 Develop your first GUI software

Since you learned to develop command-line software earlier, the steps for developing graphical interface software are the same. There are two differences, one is to replace the *app* with *app-gtk4*, and the other is that the startup method of the graphical interface software should be started through the program launcher.

- Open the program launcher and find *app-gtk4*:



- Click to open *app-gtk4*:



Congratulations, you have successfully developed a graphical interface software again.

6 Learn Chez Scheme

- To learn Scheme programming, you can refer to SICP:

Structure and Interpretation of Computer Programs

- To learn the Chez Scheme API, you can refer to the official documentation:

The Scheme Programming Language (4th Edition)
The Chez Scheme User's Guide for Version 9.5.8

- For quick query function usage, please refer to the list of functions:

Function Index

7 Learn Minipaca built-in functions

7.1 *f-e*

This function is used to store global variables in a software.

- (**f-e** *pair*) → unspecified

Put *pair* into *f-e*.

- (**f-e** *key*) → a pair or a boolean #f

Get *pair* from *f-e* by the *key*. If not found, returns #f.

7.2 *f-t*

Translation of string.

- (**f-t** *lang*) → unspecified

Set current language to *lang*. *lang* must be a symbol.

- (**f-t** *pair*) → unspecified

Put translation into *f-t*. The *key* of *pair* is a string and the *value* is the translated string.

- (**f-t** *string*) → a pair

Get translation from *f-t*. The *key* of *pair* is a string and the *value* is the translated string. If not found, the translated string if the string itself.

7.3 *f-copy-file*

- (**f-copy-file** *a b*) → unspecified

Copy file *a* to *b*.

7.4 *f-cpdir-rec*

- (**f-cpdir-rec** *m n*) → unspecified

Copy directory *m* into *n* recursively.

7.5 *f-mkdir-rec*

- (**f-mkdir-rec** *m*) → unspecified

Make directory *m* recursively.

7.6 *f-rmdir-rec*

- (**f-rmdir-rec** *m*) → unspecified

Remove directory *m* recursively.

7.7 *f-disk-usage*

- (**f-disk-usage** path) → a number

Summarize disk usage of a file or a directories recursively.

7.8 *f-which-executable-file*

- (**f-which-executable** str) → a path or #f

The full path of a command. If not found, returns #f.

7.9 *f-string-split*

- (**f-string-split** *string delim*) → a list

Split *string* by *delim*. *delim* is a character.

7.10 *f-datum-file-read*

- (**f-datum-file-read** *file*) → an alist

Read *alist* form a datum *file*.

7.11 *f-datum-file-write*

- (**f-datum-file-write** *alist file*) → unspecified

Write *alist* into a datum *file*.

8 Extend your first CLI software

- Understand the source code structure of software

Now that you're familiar with the basics of using Minipaca and the Chez Scheme programming language, it's time to try extending your first command-line software.
Before you really get started, it's worth understanding the source code structure of the default app software.

```
$ minipaca -x app
$ ls app/
distro doc source
```

distro → The folder where release information is stored
doc → The folder where the document is stored
source → The folder where the source code is stored

```
$ ls app/distro
all.datum.ss            debiant.datum.ss    fc37.datum.ss
fc38.datum.ss           ub2304.datum.ss     arch.datum.ss
debian12.datum.ss       fc36.datum.ss       ub2204.datum.ss
...
```

all.datum.ss → Software name, version number, etc.
other .datum.ss → Distro-related information, etc.

```
$ ls app/doc
log
```

log → File that records software changes

```
$ ls app/source
f i x ss.ss
```

ss.ss → Main script source file
x → The folder that holds functions starting with *x-*
i → The folder that holds functions starting with *i-*
f → The folder that holds functions starting with *f-*

Before compiling, the software will copy the f function, i function, and x function to the front of the ss.ss file. Therefore, the hierarchy of various functions is, from high to low:

ss.ss
x
i
f
Chez Scheme

The enforced rule is :(1) A function can only call functions that are lower than it; (2) A file defines only one function; (3) The x function accepts any number of arguments, and the return value type must be a string, such as:

```
(define x-name
  (lambda parameters
    ...
    "abcdef"))
```

The recommended rule is: Do not add or modify the f function, which will leave a convenient room for subsequent upgrades. Your focus should be on writing x and i functions.

- Change the software name and version number

Open the *all.datum.ss* file, change the corresponding items, and recompile and package.

- Add a feature to the software

For example, you want to read a text file and display it:

Create an new file **source/x/x-cat.ss**:

```
(define x-cat
  (lambda parameters
    (let* ((file (car parameters))
           (port (open-input-file file))
           (str (get-string-all port)))
      (close-port port)
      str)))
```

Change the file **source/ss.ss**:

```
...
  (if (and (>= (length (cdr (command-line))) 2)
           (string=? (cadr (command-line)) "-x"))
      (let ((s (caddr (command-line))))
        (case s
         ("version" (set! x "x-version"))
         ("usage" (set! x "x-usage"))
         ("cat" (set! x "x-cat"))       ;; Add this line
         (else (set! x "x-usage")))
        (set! parameters (cdddr (command-line)))))
...
```

Have a try after recompiling:

```
$ minipaca -g
$ g/XX/usr/bin/app -x ./text.txt
## XX refers to your distro
```

After multiple modifications, after achieving your purpose, you can compile, package, install and have a try.

More examples at: `https://github.com/minipaca`

9 Learn GTK4

Learning GTK4 is mainly about browsing the official documentation: GTK Documentation。The most important parts are GTK, GDK and GSK.

10 Extend your first GUI software

Looking at the source code of the *app* and *app-gtk4*, we find two more function files:

f-gtk4.ss	→	Chez Scheme's binding to GTK4
x-gtk4.ss	→	Code that defines the graphical interface

Let's not modify *f-gtk4.ss* file, just modify the *x-gtk4.ss* file. Through the previous skills, constantly modify and have a try until it meets your requirements.

11 Porting to other platforms and distros

Software developed with Minipaca is inherently cross-platform. Simply compile and package your software in the platforms you want to support.