GNU Mes Reference Manual

Full Source Bootstrapping for the GNU system

Jan (janneke) Nieuwenhuizen

Edition 0.22
20 January 2020

Copyright © 2018,2019 Jan (janneke) Nieuwenhuizen

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".

Table of Contents

GNU Mes

This document describes GNU Mes version 0.22, a bootstrappable Scheme interpreter and C compiler written for bootstrapping the GNU system.

1 Introduction

> These were "Maxwell's Equations of Software!"
> —*Alan Kay*

The purpose of GNU Mes[1] is to help create a computer operating system that we can trust.

Mes consists of a mutual self-hosting Scheme interpreter written in C and a Nyacc-based (see see Section "NYACC User's Guide" in *NYACC User's Guide*) C compiler written in Scheme. The Scheme interpreter `mes.c` is about 5,000LOC of restricted C, to be compiled with M2-Planet[2], a very simple C compiler.

If we want to trust our computers to do what we instructed them to do then we need to be able to inspect all instructions—all softwares—that we have given it to run.

1.1 Software Freedom

The four essential Freedoms of Software are at the core of our GNU community. Quoting the GNU philosophy[3]

> A program is free software if the program's users have the four essential freedoms:
>
> 0. The freedom to run the program as you wish, for any purpose (freedom 0).
> 1. The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
> 2. The freedom to redistribute copies so you can help others (freedom 2).
> 3. The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

A computer operating system that respects the user's freedom is one essential ingredient for building a reliable, trustable computing system. There are about a dozen general purpose operating systems that can be trusted in this way, see Free Distributions (`https://www.gnu.org/distros/free-distros.html`). For all softwares on such a system we have the full source code and build recipes available.

So we have access to all the software, we have studied it, possibly modified it, then we built it and we installed it on a computer or some device or appliance. How can we trust that when we run the program we are indeed running the untainted product of the source code that we studied? Unless we are certain of this we cannot really enjoy Freedom 1.

---

[1] "Mes" is an acronym for the Maxwell Equations of Software.

[2] See `https://github.com/oriansj/m2-planet`

[3] The four essential freedoms `https://www.gnu.org/philosophy/free-sw.html`

1.2 Reproducible Builds

The current Reproducible Builds effort incubated in the Debian project[4] and was organized by Lunar. Quoting the Reproducible Builds website[5]

> A build is reproducible if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.

1.2.1 Can we trust our freedom?

Now consider the opposite, that a second build of a piece of source code produces a different binary program. Upon further investigation we might find that the only difference is probably harmless: a timestamp that was embedded in the binary, or perhaps the name of the user that built it or directory it was built in. Such investigations can be nontrivial and are highly unpractical. And what if the binary difference is not so trivial, cannot be easily accounted for?

A piece of software that cannot be built bit-by-bit reproducible is probably not a good community member in the world of software freedom. We think the importance of reproducibility should not be underestimated largely because failing that precondition makes justifable trust in binaries provided suspect at best and downright dangerous in reality.

It becomes clear that a bit-by-bit reproducible build of all our sofwares is essential if we value our Freedom 1.

1.2.2 An Old Idea

The idea of reproducible builds is not very new. It was implemented for GNU tools in the early 1990s (which we learned, much later in 2017). In the Debian world it was mentioned first in 2000 and then more explicitly in 2007 on debian-devel[6]

> I think it would be really cool if the Debian policy required that packages could be rebuild bit-identical from source.
>
> —*Martin Uecker*

1.3 Bootstrappable Builds

Software distributions that take reproducible builds seriously are currently shipping well over 90% reproducible packages.

That a package builds bit-by-bit reproducibly however is not enough to guarantee Freedom 1. There is another factor that is often overlooked: opaque ascii or binary *seeds* that are injected during build time. Yes, a package may build reproduciblly from all inspectable sourcess...but what functionality is programmed in the opaque seed?

1.3.1 Bootstrap Binary Seed

Possibly one of the most harmless, but certainly by far the biggest binary seed that all software distributions inject are the so called *bootstrap binary seed*. Bootstrap binaries are the initial binary seeds that are used to start building the distribution.

[4] The Debian Project (`http://debian.org`)

[5] Reproducible Builds (`https://reproducible-builds.org/`)

[6] Martin Uecker on debian-devel on bit-reproducibility (`https://lists.debian.org/debian-devel/2007/09/msg00746.html`)

The GNU Guix operating system, version 1.0 had a relatively small closure of bootstrap binary seed: GNU binutils, GNU gcc, GNU Libc, GNU Guile, and "Static binaries" (think: bash, bzip2, coreutils, gawk, grep, gzip, patch, sed, tar, xz).

```
$ du -schx $(readlink $(guix build bootstrap-tarballs)/*)
2.1M /gnu/store/9623n4bq6iq5c8cwwdq99qb7d0xj93ym-binutils-static-stripped-tarball-2.28
18M /gnu/store/437xwygmmwwpkddcyy1qvjcv4hak89pb-gcc-stripped-tarball-5.5.0/gcc-strippe
1.8M /gnu/store/55ccx18a0d1x5y6a575jf1yr0ywizvdg-glibc-stripped-tarball-2.26.105-g0890
5.7M /gnu/store/bqf0ajclbvnbm0a46819f30804y3ilx0-guile-static-stripped-tarball-2.2.3/g
5.8M /gnu/store/j8yzjmh9sy4gbdfwjrhw46zca43aah6x-static-binaries-tarball-0/static-bina
33M total
```

only a 33MB download that unpacks to a 252MB *seed* of opaque binary code.

```
$ for i in $(readlink $(guix build bootstrap-tarballs)/*);\
  do sudo tar xf $i; done
$ du -schx *
130M bin
13M include
54M lib
51M libexec
5.2M share
252M total
```

During the Guix 1.1 development series we managed to create the first reduction by 50% of the Guix *bootstrap binary seed*[7]. This was a very important step because the ~250MB *seed* of binary code was practically non-auditable, which makes it hard to establish what source code produced them.

1.4 Full Source Bootstrap

There is an obvious solution: we cannot allow any binary seeds in our software stack. Not even in the bootstrap binary seed. Maybe that is a bit too strong: we want to have the absolute minimum of binary seeds and all binary seeds need to be inspectable and must be reviewed. How big would the absolute minimal set be?

1.4.1 The Magical Self-Hosting Hex Assembler

June 2016 I learnt about Stage0 (`https://github.com/oriansj/stage0/`). Jeremiah Orians created `hex0` a ~500 byte self-hosting hex assembler. The source code is well documented and the binary is the exact mirror of the source code. I was inspired.

Here is an example of what the `hex0` program looks like; the start of the *hex* function

```
00000060: 4883 f830 7c6f 4883 f83a 7c5a 4883 f841  H..0|oH..:|ZH..A
...
000000d0: 48c7 c0ff ffff ffc3 0000 0000 0000 0000  H...............
000000e0: 4883 e830 c300 0000 0000 0000 0000 0000  H..0............
```

All computer programs look like this: an opaque list of computer codes. The initial programs that we take for granted—the bootstrap binary seed—are about 250MB of such numbers: think 250,000 pages full of numbers.

[7] See `https://guix.gnu.org/blog/2019/guix-reduces-bootstrap-seed-by-50/`

Most computers work pretty well so apparently there is not a pressing need to inspect and study all of these codes. At the same time it is tricky to fully trust[8] a computer that was bootstrapped in this way.

Here is what the source code of the `hex0` assembler looks like

```
## function: hex
48 83 f8 30                # cmp $0x30,%rax
7c 6f                      # jl 6000f3 <ascii_other>
48 83 f8 3a                # cmp $0x3a,%rax
7c 5a                      # jl 6000e4 <ascii_num>
48 83 f8 41                # cmp $0x41,%rax
...
## function: ascii_other
48 c7 c0 ff ff ff ff       # mov $0xffffffffffffffff,%rax
c3                         # ret
...
## function: ascii_num
48 83 e8 30                # sub $0x30,%rax
c3                         # ret
```

While it may be hard to understand what this piece of the program does, it should be possible for anyone to verify that the computer codes above correspond to the source code with comments.

One step beyond these annotated codes is Assembly language. To write a program in Assembly, you only need to provide the instructions; the codes are computed by the `assembler` program.

```
hex:
# deal all ascii less than 0
cmp $48, %rax
jl ascii_other
# deal with 0-9
cmp $58, %rax
jl ascii_num
...
ascii_other:
mov $-1, %rax
ret
ascii_num:
sub $48, %rax
ret
```

More readable still, a similar program text in the C programming language.

```
int
hex (int c)
{
  if (c >= '0' && c <= '9')
```

[8] Ken Thompson's 1984 Turing award acceptance speech Reflections on Trusting Tust (`http://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf`).

```
        return c - 48;
    ...
    }
```

What if we could bootstrap our entire system from only this one `hex0` assembler binary seed? We would only ever need to inspect these 500 bytes of computer codes. Every[9] later program is written in a more friendly programming language: Assembly, C, ... Scheme.

Inspecting all these programs is a lot of work, but it can certainly be done. We might be able to create a fully inspectable path from almost nothing to all of the programs that our computer runs. Something that seemed to be an impossible dream is suddenly starting to look like "just a couple years of work".

1.5 LISP as Maxwell's Equations of Software

As fate would have it, I stumbled upon this interview with Alan Kay (`https://queue.acm.org/detail.cfm?id=1039523`), where he shares a revelation he had when reading John McCarthy's LISP-1.5 (`http://www.softwarepreservation.org/projects/LISP/book/LISP%201.5%20Programmers%20Manual.pdf`) manual:

> that was the big revelation to me ... when I finally understood that the half page of code on the bottom of page 13 of the Lisp 1.5 manual was Lisp in itself. These were "Maxwell's Equations of Software!" This is the whole world of programming in a few lines that I can put my hand over.
>
> —*Alan Kay*

Our starting point is `hex0`, a 500 byte hex assembler and we need to somehow close the gap to building the bootstrap binary seed, esp. GNU Gcc and the GNU C Library. What better way to do that than by leveraging the powers of LISP?

GNU Mes is a Scheme[10] interpreter that will be indirectly bootstrapped from `hex0` and that wields the magical powers of LISP to close the bootstrap gap, asserting we can enjoy software Freedom 1.

1.5.1 Auditable Elegance

`eval` and `apply` are mutual recursing functions that—using a few helper functions—describe the core of the universe of computing.

```
(define (apply fn x a)
  (cond
   ((atom fn)
    (cond
     ((eq fn CAR)  (caar x))
     ((eq fn CDR)  (cdar x))
     ((eq fn CONS) (cons (car x) (cadr x)))
     ((eq fn ATOM) (atom (car x)))
```

[9] Some program languages have become very hard or practically impossible to bootstrap. Instead of depending on a simple language such as C, they depend on a recent version of itself, or on other binary or ASCII seeds, on other recent programs written in that language, or even on manual intervention. Programs written in a language that cannot be bootstrapped can still run on our systems, but cannot enjoy any of the trust we intend to create.

[10] Scheme is a modern LISP

```
      ((eq fn EQ)   (eq (car x) (cadr x)))
      (#t           (apply (eval fn a) x a))))
    ((eq (car fn) LAMBDA)
                  (eval (caddr fn) (pairlis (cadr fn) x a)))
    ((eq (car fn) LABEL)
                  (apply (caddr fn) x
                         (cons (cons (cadr fn) (caddr fn)) a)))))
(define (eval e a)
  (cond
    ((atom e) (cdr (assoc e a)))
    ((atom (car e))
     (cond ((eq (car e) QUOTE) (cadr e))
           ((eq (car e) COND)  (evcon (cdr e) a))
           (#t                 (apply (car e) (evlis (cdr e) a) a))))
    (#t       (apply (car e) (evlis (cdr e) a) a))))
```

It will be a big day when our computers are fully bootstrapped from source. It would be nice if that source code were readable, auditable and elegant. To be honest, the elegance displayed above that we achieved at the very start of the Mes project is currently hard to find. It is our sincerest hope to bring back this level of quality and elegance..

2 Installation

Mes is available for download from its website at `https://www.gnu.org/pub/gnu/mes/`. This section describes the software requirements of Mes, as well as how to install it and get ready to use it.

2.1 Regular Requirements

This section lists requirements when building Mes from source. The build procedure for Mes is the same as for other GNU software, and is not covered here. Please see the files `README` and `INSTALL` in the Mes source tree for additional details.

GNU Mes depends on the following packages:

- GNU Guile (`http://gnu.org/software/guile/`), version 2.0.13 or later, including 2.2.x;
- GNU Make (`http://www.gnu.org/software/make/`).
- NYACC (`https://savannah.gnu.org/projects/nyacc/`), 0.93.0 or later, including 0.99.0.
- GCC's gcc (`http://gcc.gnu.org`), version 2.95.3 or later.
- mescc-tools (`https://savannah.gnu.org/projects/mescc-tools/`), version 0.6.1 or later,

Mes is compatible with GNU Guile, so it is possible to share the same Scheme code between both. Currently Mes only supports the minimal subset of R5RS and Guile extensions to run MesCC.

2.2 Bootstrap Requirements

This section lists requirements when building Mes as a bootstrap package. The bootstrap build procedure for Mes is similar to building GNU software and goes like this

```
sh configure.sh --prefix=/your/prefix/here
sh bootstrap.sh
sh check.sh
sh install.sh
```

See `configure.sh` and `bootstrap.sh` for inspiration on what environment variables to set.

Bootstrapping Mes depends on the following packages:

- a POSIX-compatible shell
- mescc-tools (`https://github.com/oriansj/mescc-tools/`), version 0.6.1 or later.
- NYACC (`https://savannah.gnu.org/projects/nyacc/`), 0.93.0 or later, including 0.99.0.

2.3 Running the Test Suites

After a successful `configure` and `make` run, it is a good idea to run the test suites.

```
make check
```

Run Mes Scheme language semantics tests (`scaffold/boot`) only

```
build-aux/check-boot.sh
```

Run a single Mes boot test

```
MES_BOOT=scaffold/boot/00-zero.scm bin/mes
```

Run a single Mes Scheme test

```
./pre-inst-env tests/boot.test
MES=guile ./pre-inst-env tests/boot.test
```

Run MesCC tests only

```
build-aux/check-mescc.sh
```

Run a single MesCC test

```
CC=gcc CC32=i686-unknown-linux-gnu-gcc MES=guile \
  build-aux/test.sh scaffold/tests/00-exit-0
```

3 Bootstrapping

> Recipe for yogurt: Add yogurt to milk.
> —*Anonymous*

The bootstrap problem we have set out to solve is that none of our modern software distributions, and Guix in particular, can be created all from source code. In addition to the carefully signed source code of all the programs (the 'milk') an opaque binary seed (the 'yogurt') is injected as an essential dependency.

Why would this be a problem, I hear you ask? This is how it is done, we always did it this way, everyone does it like this! Indeed, a popular way of handling the bootstrapping issue is by ignoring it.

> Your compiler becoming self-hosting. . . a language creator's wet dream.
> —*PFH*

It seems that writing a self-hosting compiler is considered to be a language creator's ultimate goal. It means that their language and compiler have become powerful enough to not depend on a pre-exising language that possibly is—but certainly was until now—more powerful; it feels like passing the rite to adulthood.

When you see the irony, you grasp what our bootstrapping effort means in practice. Creating bootstrappable software is not hard; actually most softwares' first releases are bootstrappable. The problem of bootstrapping is not a technical one, it is a lack of awareness and responsibility.

3.1 The Mes Bootstrap Process

The Reduced Binary Seed bootstrap currently adopted by Guix[1]. In its intiial form it is only available for x86-linux.

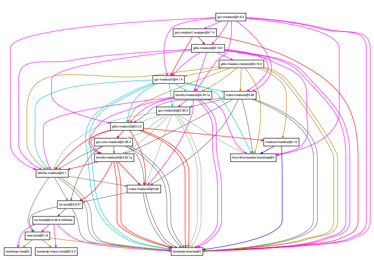
Currently, it goes like this:

```
 gcc-mesboot (4.9.4)
        ^
        |
glibc-mesboot (2.16.0)
        ^
        |
 gcc-mesboot1 (4.7.4)
        ^
        |
binutils-mesboot (2.20.1a)
        ^
        |
 gcc-mesboot0 (2.95.3)
        ^
        |
 glibc-mesboot0 (2.2.5)
```

[1] See `gnu/packages/commencement.scm` in the *master* branch in Guix git `http://git.savannah.gnu.org/cgit/guix.git/tree/gnu/packages/commencement.scm`

```
                              ^
                              |
                      gcc-core-mesboot (2.95.3)
                              ^
                              |
   make-mesboot0, diffutils-mesboot, binutils-mesboot0 (2.20.1a)
                              ^
                              |
                          tcc-boot
                              ^
                              |
                          tcc-boot0
                              ^
                              |
                          mes-boot
                              ^
                              |
                              *
             bootstrap-mescc-tools, bootstrap-mes (~10MB)
     bootstrap-bash, bootstrap-coreutils&co, bootstrap-guile (~120MB)
```

Here's a generated dependency diagram to for the final bootstrap gcc that builds the rest of Guix.



Work is ongoing to remove these binary seeds that were intentionally injected by our own doing as temporary shortcut

```
bootstrap-mescc-tools (seed), bootstrap-mes (seed)
```

For now, these additional non-bootstrapped dependencies (i.e., binary seeds) are taken for granted

```
bootstrap-guile, bash, bzip2, coreutils, gawk, grep, gzip, patch, sed,
tar, xz
```

Although we think these are less essential and thus less interesting than the GNU toolchain triplet that we focussed on initially, our next priority is to eleminate these one by one.

3.2 Invoking mes

The `mes` command is the Scheme interpreter whose prime directive is to run the `MesCC` program.

For convenience and testing purposes, `mes` tries to mimic guile.

```
mes option... FILE...
```

The *options* can be among the following:

`-s` *script arg...*
: By default, mes will read a file named on the command line as a script. Any command-line arguments *arg*... following *script* become the script's arguments; the `command-line` function returns a list of strings of the form `(script arg...)`.

  Scripts are read and evaluated as Scheme source code just as the `load` function would. After loading *script*, mes exits.

`-c` *expr arg...*
: Evaluate *expr* as Scheme code, and then exit. Any command-line arguments *arg*...) following *expr* become command-line arguments; the `command-line` function returns a list of strings of the form `(guile arg...)`, where *mes* is the path of the mes executable.

`--` *arg...*
: Run interactively, prompting the user for expressions and evaluating them. Any command-line arguments *arg*... following the `--` become command-line arguments for the interactive session; the `command-line` function returns a list of strings of the form `(guile arg...)`, where *mes* is the path of the mes executable.

`-L,--load-path=`*directory*
: Add *directory* to the front of Mes module load path. The given directories are searched in the order given on the command line and before any directories in the `GUILE_LOAD_PATH` environment variable.

`-C,--compiled-path=`*directory*
: Accepted and ignored for Guile compatibility.

`-l` *file*
: Load Scheme source code from *file*, and continue processing the command line.

`-e,--main=`*function*
: Make *function* the *entry point* of the script. After loading the script file (with `-s`) or evaluating the expression (with `-c`), apply *function* to a list containing the program name and the command-line arguments—the list provided by the `command-line` function.

`-h`, `--help`
: Display help on invoking mes, and then exit.

`-v`, `--version`
: Display the current version of mes%, and then exit.

3.2.1 Environment Variables

Here are the environment variables (see see Section "Environment Variables" in *Guile Reference*) that affect the run-time behavior of mes:

`MES_BOOT`
: Set `MES_BOOT` to change the initial Scheme program that mes runs.

`MES_ARENA`
: The initial size of the arena see Section "5.3" in *SICP* in cells. Default: 20,000.

`MES_MAX_ARENA`
: The maximum size of the arena in cells. Default: 100,000,000.

`MES_MAX_STRING`
: The maximum size of a string. Default: 524,288.

`MES_DEBUG`

1. Informational:
   - MODULEDIR
   - included SCM modules and sources
   - result of program
   - gc stats at exit
2. opened files
3. runtime gc stats
4. detailed info
   - parsed, expanded program
   - list of builtins
   - list of symbol
   - opened input strings
   - gc details
5. usage of opened input strings

`GUILE_LOAD_PATH`
: This variable may be used to augment the path that is searched for Scheme files when loading. Its value should be a colon-separated list of directories. If it contains the special path component `...` (ellipsis), then the default path is put in place of the ellipsis, otherwise the default path is placed at the end. The result is stored in `%load-path`.

Mes uses **GUILE**_*LOAD_PATH* for compatibility with Guile.

3.3 Invoking mescc

```
mescc option... FILE...
```

The *option*s can be among the following:

`--align`
: align globals

`--base-address=ADRRESS`
: use BaseAddress ADDRESS [0x1000000]

`-c`
: preprocess, compile and assemble only; do not link

`-D DEFINE[=VALUE]`

`-dumpmachine`
: display the compiler's target processor

`-E` preprocess only; do not compile, assemble or link

`-g` add `blood-elf` debug info

This enables GDB setting breakpoints on function names, and to have the GDB backtrace command to show the function call stack.

`-h, --help`
display this help and exit

`-I DIR` append DIR to include path

`-L DIR` append DIR to library path

`-l LIBNAME`
link with LIBNAME

`-m BITS` compile for BITS bits [32]

`-O LEVEL` use optimizing LEVEL

`-o FILE` write output to FILE

`-S` preprocess and compile only; do not assemble or link

`--std=STANDARD`
assume that the input sources are for STANDARD

`-V,--version`
display version and exit

`-w,--write=TYPE`
dump Nyacc AST using TYPE {pretty-print,write}

`-x LANGUAGE`
specify LANGUAGE of the following input files

3.3.1 MesCC Environment Variables

`MES`

Setting `MES` to a mes-compatible Scheme will run mescc using that

```
MES=guile mescc -c scaffold/main.c
```

See, now Guile has become compatible with Mes, instead of the other way around ;-)

`C_INCLUDE_PATH`
`LIBRARY_PATH`
`NYACC_DEBUG`
Setting `NYACC_DEBUG` makes nyacc print names of function during the parsing phase.

3.4 Invoking mesar

```
mesar option... command ARCHIVE-FILE FILE...
```

The *command* is ignored for compatibility with `ar`

```
r[ab][f][u]  - replace existing or insert new file(s) into the archive
[c]          - do not warn if the library had to be created
[D]          - use zero for timestamps and uids/gids (default)
```

and assumed to be *crD*.

The *option*s can be among the following:

`-h, --help`
: display this help and exit

`-V,--version`
: display version and exit

4 Contributing

4.1 Building from Git

If you want to hack GNU Mes itself, it is recommended to use the latest version from the Git repository:

```
git clone git://git.savannah.gnu.org/mes.git
```

The easiest way to set up a development environment for Mes is, of course, by using Guix! The following command starts a new shell where all the dependencies and appropriate environment variables are set up to hack on Mes:

```
guix environment -l .guix.scm
```

Finally, you have to invoke `make check` to run tests (see Section 2.3 [Running the Test Suites], page 9). If anything fails, take a look at installation instructions (see Chapter 2 [Installation], page 8) or send a message to the `bug-mes@gnu.org` mailing list.

4.2 Running Mes From the Source Tree

First, you need to have an environment with all the dependencies available (see Section 4.1 [Building from Git], page 16), and then simply prefix each command by `./pre-inst-env` (the `pre-inst-env` script lives in the top build tree of Mes).

4.3 Porting GNU Mes

Mes was written for x86-linux. A 64 bit (x86_64) is almost done, only a few bugs remain. The Guix bootstrap for x86_64 uses x86 mes and that is not expected to change.

An ARM (armv4/armv7l) linux port is underway. A port to GNU/Hurd (x86-gnu) is also underway.

Initial scaffold, built by `build-aux/build-scaffold.sh`:

```
lib/linux/x86-mes-gcc/exit-42.S
lib/linux/x86-mes/elf32-0exit-42.hex2
lib/linux/x86-mes/elf32-body-exit-42.hex2

lib/linux/x86-mes-gcc/hello-mes.S
lib/linux/x86-mes/elf32-0hello-mes.hex2
lib/linux/x86-mes/elf32-body-hello-mes.hex2
```

Porting MesCC:

```
lib/x86-mes/x86.M1

module/mescc/mescc.scm
module/mescc/i386/as.scm
module/mescc/i386/info.scm

mes/module/mescc/i386/as.mes
mes/module/mescc/i386/info.mes
```

4.4 The Perfect Setup

The Perfect Setup to hack on Mes is basically the perfect setup used for Guile hacking (see Section "Using Guile in Emacs" in *Guile Reference Manual*). First, you need more than an editor, you need Emacs (`http://www.gnu.org/software/emacs`), empowered by the wonderful Geiser (`http://nongnu.org/geiser/`).

Geiser allows for interactive and incremental development from within Emacs: code compilation and evaluation from within buffers, access to on-line documentation (docstrings), context-sensitive completion, `M-.` to jump to an object definition, a REPL to try out your code, and more (see Section "Introduction" in *Geiser User Manual*).

4.5 Coding Style

In general our code follows the GNU Coding Standards (see *GNU Coding Standards*). However, they do not say much about Scheme, so here are some additional rules.

4.5.1 Programming Paradigm

Scheme code in Mes is written in a purely functional style.

4.5.2 Formatting Code

When writing Scheme code, we follow common wisdom among Scheme programmers. In general, we follow the Riastradh's Lisp Style Rules (`http://mumble.net/~campbell/scheme/style.txt`). This document happens to describe the conventions mostly used in Guile's code too. It is very thoughtful and well written, so please do read it.

If you do not use Emacs, please make sure to let your editor knows these rules.

Additionally, in Mes we prefer to format `if` statements like this

```
(if foo? trivial-then
    (let ((bar (the-longer ...)))
      more-complicated
      ...
      else))
```

4.6 Submitting Patches

Development is done using the Git distributed version control system. Thus, access to the repository is not strictly necessary. We welcome contributions in the form of patches as produced by `git format-patch` sent to the `guix-patches@gnu.org` mailing list.

Please write commit logs in the ChangeLog format (see Section "Change Logs" in *GNU Coding Standards*); you can check the commit history for examples.

4.6.1 Reporting Bugs

Encountering a problem or bug can be very frustrating for you as a user or potential contributor. For us as Mes maintainers, the preferred bug report includes a beautiful and tested patch that we can integrate without any effort.

However, please don't let our preference stop you from reporting a bug. There's one thing *much* worse for us than getting a bug report without a patch: Reading a complaint or rant online about your frustrations and how our work sucks, without having heard directly what you experienced.

So if you report a problem, will it be fixed? And **when**? The most honest answer is: It depends. Let's curry that informationless honesty with a more helpful and more blunt reminder of a mantra of free software:

Q: When will it be finished?

A: It will be ready sooner if you help.

—*Richard Stallman*

Join us on `#bootstrappable` on the Freenode IRC network or on `guix-devel@gnu.org` to share your experience—good or bad.

Please send bug reports with full details to `bug-mes@gnu.org`.

5 Acknowledgments

We would like to thank the following people for their help: Jeremiah Orians, Peter de Wachter, rain1, Ricardo Wurmus, Rutger van Beusekom.

We also thank Ludovic Courtès for creating GNU Guix and making the bootstrap problem so painfully visible, John McCarthy for creating LISP-1.5 and Alan Kay for their inspiring comment on Page 13 (`https://queue.acm.org/detail.cfm?id=1039523`).

6 Resources

- Bootstrappable Builds (`https://bootstrappable.org`) Minimize the amount and size of opaque binary seeds we need to swallow.
- Reproducible Builds (`https://reproducible-builds.org`) Provide a verifiable path from source code to binary.
- Stage0 (`https://gitlab.com/oriansj/stage0`) If we want, it could all start with a ~500 byte self-hosting hex assembler.
- Bootstrapping wiki (`https://bootstrapping.miraheze.org`) An amazing collection of small/bootstrappable compilers, operating systems, anything you need.
- #bootstrappable (`irc.freenode.net`) The bootstrapping community home at the freenode IRC network.
- `guix-devel@gnu.org` The Guix mailing list, where it all started. guix-devel archives (`https://lists.gnu.org/archive/html/guix-devel/`).

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

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 noncommercially. 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.

Concept Index

Programming Index

(Index is nonexistent)