guile-r7rs

sourcehut success

Introduction

guile-r7rs is the collection of libraries part of R7RS bundled for GNU Guile 2.2 or later.

How to contribute

1. Create an account on sr.ht. To contribute to existing repository, it is free.
2. Pick a library from the table of content (see below), check nobody is working on it in the todo.
3. Add documentation, tests or an implementation based on existing Guile modules or sample implementation that can be found at http://srfi.schemers.org/.
4. When your contribution is ready, ask amirouche at hyper dev to become a contributor to be able to push.

Don't forget to add your name in the license header.

When you add a documentation file, don't forget to add it to `DOCUMENTATION_FILES` inside the `Makefile`. To build the documentation you will need `pandoc`, `latex` and to run `make doc`.

When you add a test file, don't forget to add it to `TESTS_FILES` inside `Makefile`. To run the tests use `make check`.

Table of Content

R7RS small

- `(scheme base)`
- `(scheme case-lambda)`
- `(scheme char)`
- `(scheme complex)`
- `(scheme cxr)`
- `(scheme eval)`
- `(scheme file)`
- `(scheme inexact)`
- `(scheme lazy)`

- `(scheme load)`
- `(scheme process-context)`
- `(scheme r5rs)`
- `(scheme read)`
- `(scheme repl)`
- `(scheme time)`
- `(scheme write)`

R7RS Red Edition

- `(scheme box)` aka. SRFI 111
- `(scheme charset)` aka. SRFI 14
- `(scheme comparator)` aka. SRFI 128
- `(scheme ephemeron)`) aka. SRFI 124
- `(scheme hash-table)` aka. SRFI 125
- `(scheme ideque)`) aka. SRFI 134
- `(scheme ilist)` aka. SRFI 116
- `(scheme list)` aka. SRFI 1
- `(scheme list-queue)` aka. SRFI 117
- `(scheme lseq)` aka. SRFI 127
- `(scheme rlist)` aka SRFI 101
- `(scheme set)` aka. SRFI 113
- `(scheme sort)` aka. SRFI 132
- `(scheme stream)` aka. SRFI 41
- `(scheme text)` aka. SRFI 135
- `(scheme vector)` aka. SRFI 133

R7RS Tangerine Edition

- `(scheme mapping)` aka. SRFI 146
- `(scheme mapping hash)` aka. SRFI 146
- `(scheme regex)` aka. SRFI 115
- `(scheme generator)` aka. SRFI 158
- `(scheme division)` aka. SRFI 141
- `(scheme bitwise)` aka. SRFI 151
- `(scheme fixnum)` aka. SRFI 143
- `(scheme flonum)` aka. SRFI 144
- `(scheme bytevector)` aka. `(rnrs bytevectors)` aka. SRFI 4
- `(scheme vector @)` aka. SRFI 160 where @ is any of base, u8, s8, u16, s16, u32, s32, u64, s64, f32, f64, c64, c128.
- `(scheme show)` aka. SRFI 159

`(scheme base)`

`_`

TODO (missing in r7rs?)

`...`

It is called ellipsis. It used in macros, `match` that is not part of R7RS. It signify that a pattern must be repeated.

`=>`

TODO

`else`

Used in `cond` form as in the last clause as a fallback.

`(* number ...)`

Multiplication procedure.

`(+ number ...)`

Addition procedure.

`(- number ...)`

Substraction procedure.

`(/ number number ...)`

Division procedure. Raise `'numerical-overflow` condition in case where denominator is zero.

`(< number number ...)`

Less than procedure. Return a boolean.

`(<= number number ...)`

Less than or equal procedure. Return a boolean.

`(= number number ...)`

Return `#t` if the numbers passed as parameters are equal. And `#f` otherwise.

`>`

Greater than procedure. Return boolean.

`>=`

TODO

`abs`

TODO

`and`

TODO

`append`

TODO

`apply`

TODO

`assoc`

TODO

`assq`

TODO

`assv`

TODO

`begin`

TODO

`binary-port?`

TODO

`boolean=?`

TODO

`boolean?`

TODO

`bytevector`

TODO

`bytevector-append`

TODO

`bytevector-copy`

TODO

`bytevector-copy!`

TODO

`bytevector-length`

TODO

`bytevector-u8-ref`

TODO

`bytevector-u8-set!`

TODO

`bytevector?`

TODO

`caar`

TODO

`cadr`

TODO

`call-with-current-continuation`

TODO

`call-with-port`

TODO

`call-with-values`

TODO

`call/cc`

TODO

`car`

TODO

`case`

TODO

`cdar`

TODO

`cddr`

TODO

`cdr`

TODO

`ceiling`

TODO

`char->integer`

TODO

`char-ready?`

TODO

`char<=?`

TODO

`char<?`

TODO

`char=?`

TODO

`char>=?`

TODO

`char>?`

TODO

`char?`

TODO

`close-input-port`

TODO

`close-output-port`

TODO

`close-port`

TODO

`complex?`

TODO

`cond`

TODO

`cond-expand`

TODO

`cons`

TODO

`current-error-port`

TODO

`current-input-port`

TODO

`current-output-port`

TODO

`define`

TODO

`define-record-type`

TODO

`define-syntax`

TODO

`define-values`

TODO

`denominator`

TODO

`do`

TODO

`dynamic-wind`

TODO

`eof-object`

TODO

`eof-object?`

TODO

`eq?`

TODO

`equal?`

TODO

`eqv?`

TODO

`(error [who] message . irritants)`

Raise an error.

`error-object-irritants`

TODO

`error-object-message`

TODO

`error-object?`

TODO

`even?`

TODO

`exact`

TODO

`exact-integer-sqrt`

TODO

`exact-integer?`

TODO

`exact?`

TODO

`expt`

TODO

`features`

TODO

`file-error?`

TODO

`floor`

TODO

`floor-quotient`

TODO

`floor-remainder`

TODO

`floor/`

TODO

`flush-output-port`

TODO

`for-each`

TODO

`gcd`

TODO

`get-output-bytevector`

TODO

`get-output-string`

TODO

`guard`

TODO

`if`

TODO

`include`

TODO

`include-ci`

TODO

`inexact`

TODO

`inexact?`

TODO

`input-port-open?`

TODO

`input-port?`

TODO

`integer->char`

TODO

`integer?`

TODO

`lambda`

TODO

`lcm`

TODO

`length`

TODO

`let`

TODO

`let*`

TODO

`let*-values`

TODO

`let-syntax`

TODO

`let-values`

TODO

`letrec`

TODO

`letrec*`

TODO

`letrec-syntax`

TODO

`list`

TODO

`list->string`

TODO

`list->vector`

TODO

`list-copy`

TODO

`list-ref`

TODO

`list-set!`

TODO

`list-tail`

TODO

`list?`

TODO

`make-bytevector`

TODO

`make-list`

TODO

`make-parameter`

TODO

`make-string`

TODO

`make-vector`

TODO

`map`

TODO

`max`

TODO

`member`

TODO

`memq`

TODO

`memv`

TODO

`min`

TODO

`modulo`

TODO

`negative?`

TODO

`newline`

TODO

`not`

TODO

`null?`

TODO

`number->string`

TODO

`number?`

TODO

`numerator`

TODO

`odd?`

TODO

`open-input-bytevector`

TODO

`open-input-string`

TODO

`open-output-bytevector`

TODO

`open-output-string`

TODO

`or`

TODO

`output-port-open?`

TODO

`output-port?`

TODO

`pair?`

TODO

`parameterize`

TODO

`peek-char`

TODO

`peek-u8`

TODO

`port?`

TODO

`positive?`

TODO

`procedure?`

TODO

`quasiquote`

TODO

`quote`

TODO

`quotient`

TODO

`raise`

TODO

`raise-continuable`

TODO

`rational?`

TODO

`rationalize`

TODO

`read-bytevector`

TODO

`read-bytevector!`

TODO

`read-char`

TODO

`read-error?`

TODO

`read-line`

TODO

`read-string`

TODO

`read-u8`

TODO

`real?`

TODO

`remainder`

TODO

`reverse`

TODO

`round`

TODO

`set!`

TODO

`set-car!`

TODO

`set-cdr!`

TODO

`square`

TODO

`string`

TODO

`string->list`

TODO

`string->number`

TODO

`string->symbol`

TODO

`string->utf8`

TODO

`string->vector`

TODO

`string-append`

TODO

`string-copy`

TODO

`string-copy!`

TODO

`string-fill!`

TODO

`string-for-each`

TODO

`string-length`

TODO

`string-map`

TODO

`string-ref`

TODO

`string-set!`

TODO

`string<=?`

TODO

`string<?`

TODO

`string=?`

TODO

`string>=?`

TODO

`string>?`

TODO

`string?`

TODO

`substring`

TODO

`symbol->string`

TODO

`symbol=?`

TODO

`symbol?`

TODO

`syntax-error`

TODO

`syntax-rules`

TODO

`textual-port?`

TODO

`truncate`

TODO

`truncate-quotient`

TODO

`truncate-remainder`

TODO

`truncate/`

TODO

`u8-ready?`

TODO

`unless`

TODO

`unquote`

TODO

`unquote-splicing`

TODO

`utf8->string`

TODO

`values`

TODO

`vector`

TODO

`vector->list`

TODO

`vector->string`

TODO

`vector-append`

TODO

`vector-copy`

TODO

`vector-copy!`

TODO

`vector-fill!`

TODO

`vector-for-each`

TODO

`vector-length`

TODO

`vector-map`

TODO

`vector-ref`

TODO

`vector-set!`

TODO

`vector?`

TODO

`when`

TODO

`with-exception-handler`

TODO

`write-bytevector`

TODO

`write-char`

TODO

`write-string`

TODO

`write-u8`

TODO

`zero?`

TODO