foreign-c a portable foreign function interface for R7RS Schemes

foreign-c

foreign-c is a C foreign function interface (FFI) library for R7RS Schemes. It is portable in the sense that it supports multiple implementations, as opposed to being portable by conforming to some specification.

Issue tracker

Maling lists

Implementation support tables

Primitives 1 table

	c-size-of	c-bytevector-u8-set!	c-byt
Chibi	X	X	
Chicken	X	X	
Gauche	X	X	
Guile	X	X	
Kawa	X	X	
Mosh	X	X	
Racket	X	X	
Saggittarius	X	X	
Stklos	X	X	
Ypsilon	X	X	

Primitives 2 table

Chibi

Chicken
Gauche
Guile
Kawa
Mosh
Racket
Saggittarius
Stklos
Ypsilon

Test files pass

	primitives.scm
Chibi	X
Chicken	X
Gauche	X
Guile	X
Kawa	X
Mosh	X
Racket	X
Saggittarius	X
Stklos	X
Ypsilon	X

Installation

Either download the latest release from https://git.sr.ht/~retropikzel/foreign-c/refs or git clone , preferably with a tag, and copy the *foreign* directory to your library directory.

Example assuming libraries in directory *snow*:

```
git clone https://git.sr.ht/~retropikzel/foreign-c --branch
LATEST_VERSION
mkdir -p snow
cp -r foreign-c/foreign snow/
make -C snow/foreign/c SCHEME_IMPLEMENTATION_NAME
```

With most implementations the make command does not compile anything. When that is the case it will say “Nothing to build on SCHEME_IMPLEMENTATION_NAME.”

Documentation

Types

Types are given as symbols, for example ’int8 or ’pointer.

- int8
- uint8
- int16
- uint16
- int32
- uint32
- int64
- uint64
- char
- unsigned-char
- short
- unsigned-short
- int
- unsigned-int
- long
- unsigned-long
- float
- double
- pointer
    - c-bytevector on Scheme side
- callback
    - Callback function
- void
    - Can not be argument type, only return type

Primitives 1

(**c-type-size** *type*)

Returns the size of given C type.

(**define-c-library** *scheme-name headers object-name options*)

Takes a scheme-name to bind the library to, list of C headers as strings, shared-object name and options.

The C header strings should not contain “<” or “>”, they are added automatically.

The name of the shared object should not contain suffix like .so or .dll. Nor should it contain any prefix like “lib”.

Options:

- additional-versions
  - Search for additional versions of shared object, given shared object “c” and additional versions “6” “7” on linux the files “libc”, “libc.6”, “libc.7” are searched for.
  - Can be either numbers or strings
- additional-paths
  - Give additional paths to search shared objects from

Example:

```
(cond-expand
  (windows (define-c-library libc-stdlib
                               '("stdlib.h")
                               "ucrtbase"
                               '((additional-versions ("0" "6"))
                                 (additiona-paths (".")))))
  (else (define-c-library libc-stdlib
                          (list "stdlib.h")
                          "c"
                          '((additional-versions ("0" "6"))
                            (additiona-paths (".")))))))
```

Notes

- Do not cond-expand inside the arguments, that might lead to problems on some implementations.
- Do not store options in variables, that might lead to problems on some implementations.
- Pass the headers using quote
  - As ’(...) and not (list...)
- Pass the options using quote
  - As ’(...) and not (list...)

(**define-c-procedure** *scheme-name shared-object c-name return-type argument-type*)

Takes a scheme-name to bind the C procedure to, shared-object where the function is looked from, c-name of the function as symbol, return-type and argument-types.

Defines a new foreign function to be used from Scheme code.

Example:

```
(cond-expand
    (windows (define-c-library libc-stdlib '("stdlib.h")
"ucrtbase" '()))
    (else (define-c-library libc-stdlib '("stdlib.h")  "c"
```

```
'("6"))))
(define-c-procedure c-puts libc-stdlib 'puts 'int '(pointer))
(c-puts "Message brought to you by foreign-c!")
```

Notes

- Pass the return-types using quote
  - As ’(…) and not (list…)

(**c-bytevector?** *obj*)

Returns **#t** if *obj* is c-bytevector, otherwise returns **#f**.

(**c-bytevector-u8-set!** *c-bytevector k byte*)

If K is not a valid index of c-bytevector the behaviour is undefined.

Stores the byte in element k of c-bytevector.

(**c-bytevector-u8-ref** *c-bytevector k*)

If K is not a valid index of c-bytevector the behaviour is undefined.

Returns the byte at index k of c-bytevector.

(**c-bytevector-pointer-set!** *c-bytevector k pointer*)

If K is not a valid index of c-bytevector the behaviour is undefined.

Stores the pointer(which is also c-bytevector) in element k of c-bytevector.

(**c-bytevector-pointer-ref** *c-bytevector k pointer*)

If K is not a valid index of c-bytevector the behaviour is undefined.

Returns the pointer(which is also c-bytevector) at index k of c-bytevector.

Primitives 2

(**define-c-callback** *scheme-name return-type argument-types procedure*)

Takes scheme-name to bind the Scheme procedure to, return-type, argument-types and procedure as in place lambda.

Defines a new Sceme function to be used as callback to C code.

Example:

```
; Load the shared library
(cond-expand
    (windows (define-c-library libc-stdlib '("stdlib.h")
"ucrtbase" '()))
```

```
    (else (define-c-library '("stdlib.h") "c" '("" "6"))))

; Define C function that takes a callback
(define-c-procedure qsort libc-stdlib 'qsort 'void '(pointer int
int callback))

; Define our callback
(pffi-define-callback compare
                      'int
                      '(pointer pointer)
                      (lambda (pointer-a pointer-b)
                        (let ((a (pffi-pointer-get pointer-a 'int
0))
                              (b (pffi-pointer-get pointer-b 'int
0)))
                          (cond ((> a b) 1)
                                ((= a b) 0)
                                ((< a b) -1)))))

; Create new array of ints to be sorted
(define array (make-c-bytevector (* (c-size-of 'int) 3)))
(pffi-pointer-set! array 'int (* (c-size-of 'int) 0) 3)
(pffi-pointer-set! array 'int (* (c-size-of 'int) 1) 2)
(pffi-pointer-set! array 'int (* (c-size-of 'int) 2) 1)

(display array)
(newline)
;> (3 2 1)

; Sort the array
(qsort array 3 (c-size-of 'int) compare)

(display array)
(newline)
;> (1 2 3)
```

c-bytevector

(**make-c-bytevector** *k*) (**make-c-bytevector** *k fill*)

Returns a newly allocated c-bytevector of *k* bytes.

If the *fill* argument is missing, the initial contents of the returned c-bytevector are unspecified.

If the *fill* argument is present, it's value must confine to C uint8_t values , it specifies the initial value for the bytes of the c-bytevector

(**make-c-null**)

Returns a null C pointer.

(**c-null?** *obj*)

Returns **#t** if *obj* is a null C pointer, otherwise returns **#f**.

(**c-free** *c-bytevector*)

Frees *c-bytevector* from memory.

native-endianness c-bytevector-s8-set! c-bytevector-s8-ref c-bytevector-s16-set! c-bytevector-s16-ref c-bytevector-s16-native-set! c-bytevector-s16-native-ref c-bytevector-u16-set! c-bytevector-u16-ref c-bytevector-u16-native-set! c-bytevector-u16-native-ref c-bytevector-s32-set! c-bytevector-s32-ref c-bytevector-s32-native-set! c-bytevector-s32-native-ref c-bytevector-u32-set! c-bytevector-u32-ref c-bytevector-u32-native-set! c-bytevector-u32-native-ref c-bytevector-s64-set! c-bytevector-s64-ref c-bytevector-s64-native-set! c-bytevector-s64-native-ref c-bytevector-u64-set! c-bytevector-u64-ref c-bytevector-u64-native-set! c-bytevector-u64-native-ref c-bytevector-sint-set! c-bytevector-sint-ref c-bytevector-uint-set! c-bytevector-uint-ref c-bytevector-ieee-single-set! c-bytevector-ieee-single-native-set! c-bytevector-ieee-single-ref c-bytevector-ieee-single-native-ref c-bytevector-ieee-double-set! c-bytevector-ieee-double-native-set! c-bytevector-ieee-double-ref c-bytevector-ieee-double-native-ref bytevector->c-bytevector c-bytevector->bytevector call-with-address-of

string->c-utf8 c-utf8->string

Environment variables

Setting environment variables like this on Windows works for this library:

```
set "PFFI_LOAD_PATH=C:\Program Files (x86)/foo/bar"
```

PFFI_LOAD_PATH

To add more paths to where pffi looks for libraries set PFFI_LOAD_PATH to paths separated by ; on windows, and : on other operating systems.