(foreign c) a portable foreign function interface for R7RS

(foreign c)

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

The new readme is a work in progress.

Implementation support tables

Primitives 1

	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

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 releases page or git clone , preferably with a tag, and copy the “foreign” directory to your library directory.

As an example assuming you have a project and your libraries live in directory called snow in it:

```
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
- callback
    - Callback function

Primitives

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

Returns the size of given C type.

(**define-c-library** *scheme-name headers object-name options*) define-c-procedure define-c-callback c-bytevector? c-bytevector-u8-set! c-bytevector-u8-ref c-bytevector-pointer-set! c-bytevector-pointer-ref

c-bytevector

make-c-bytevector make-c-null c-null? c-free 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.