Mu Namespace

for libmu version *0.0.7*

Types

T	type superclass
`:char`	*char*
`:cons`	*cons*
`:except`	*exception*
`:double`	64 bit IEEE *float*
`:fixnum`	59 bit signed *integer*
`:float`	32 bit IEEE *float*
`:func`	*function*
`:ns`	*symbol* bindings
`:stream`	file, string, socket
`:symbol`	LISP-1 binding
`:vector`	`:t` `:byte` `:char` `:fixnum` `:float`

env namespace

context capture current context
bind *fn ctx* make context composable
suspend *ctx* suspend context
resume *ctx* resume context
fix *fn* fixpoint function

cx-info the current context
ev-info the current environment
hp-info the current heap

fnv-lex *fn* capture lexical env for *fn*
fnv-set *vec fix T* bind lexical in *fn*

lex-pop *fn* pop lexical bindings
lex-psh *fn* push lexical bindings
lex-set *fn fix T* bind lexical in context
lex-ref *fix fix'* ref lexical variable of frame

ns *ns* register namespace
find-ns *string* map *string* to namespace

Symbols

boundp *symbol* is *symbol* bound?

keysymp *symbol* *keyword* predicate
keyword *string* make *keyword* of *string*

symbol *string* uninterned *symbol*
sy-name *symbol* *symbol* name binding
sy-val *symbol* *symbol* value binding
sy-ns *symbol* *symbol ns* binding

Special Forms (keywords)

:lambda *list* **.** *body* define anonymous *function*
:quote *T* quote form
:if *T T' T"* conditional

Core

eval *T* evaluate form
eq *T T'* are *T* and *T'* identical?
type-of *T* type *keyword*
funcall *fn list* apply *function* to arg *list*
gc *bool* garbage collection

version version *string* symbol

::compile *T* library form compiler
::view *T* view vector of object

::fclone *fn vector* clone *function*
::saveimg *string* save heap image to file

::fn-int *key fn* *function* internal by keyword
::hp-info *heap :keyword* heap occupancy for type

Floats

float* *float float'* product of *float* and *float'*
float+ *float float'* sum of *float* and *float'*
float- *float F'* difference of *float* and *float'*
float< *float float'* is *float* less than *float'*?
float/ *float float'* *float* divided by *float'*

Exceptions

except *tag class type T* make *exception keywords*
raise *exception* raise type *exception*
with-ex *fn fn'* catch *exception*

Lists

car *list* head of *list*
cdr *list* tail of *list*

cons *T T'* *cons* from *T* and *T'*

length *list* length of *list*

nth *fix list* nth *car* of *list*
nthcdr *fix list* nth *cdr* of *list*

Fixnums

fixnum* *fix fix'* product of *fix* and *fix'*
fixnum+ *fix fix'* sum of *fix* and *fix'*
fixnum- *fix fix'* difference of *fix* and *fix'*
fixnum< *fix fix'* is *fix* less than *fix'*?

trunc *fix fix'* truncate *fix / fix'*
floor *fix fix'* floor *fix* and *fix'*
logand *fix fix'* bitwise and of *fix* and *fix'*
logor *fix fix'* bitwise or *fix* and *fix'*

Reader

read *stream* read object from *stream*

Printer

write *T stream bool* print with escapes

Chars

code-ch *char* return *fixnum* of *char*
ch-code *fixnum* return *char* of *fixnum*

Vectors

slice *vector fix fix'* slice *vector offset length*

sv-len *vector* *fixnum* length of *vector*
sv-ref *vector fix* *nth* element
sv-type *vector* type of *vector* elements
list-sv *type list* specialized vector from list

Streams

std-in standard input *stream* symbol
std-out standard output *stream* symbol
err-out standard error *stream* symbol

close *stream* close *stream*
eofp *stream* is *stream* at end of file?
get-str *stream* get *string* from *stream*
open `type dir` *string* open *stream* from
`type` **`:file`** | **`:string`**
`dir` **`:input`** | **`:output`**

rd-byte *stream* read *byte* from *stream*
un-byte *byte stream* push *byte* onto *stream*
wr-byte *byte stream* write *byte* to *stream*
rd-char *stream* read *char* from *stream*
un-char *byte stream*
push char onto *stream*
wr-char *byte stream*
write *char* to *stream*

Namespaces

intern *ns :key string*
intern unbound symbol
`:intern` | **`:extern`**
:intern *ns* ***:****key string T*
intern value in *namespace*
ns-find *ns :key string*
map *string* to *symbol*
ns *string ns* make *namespace*
ns-imp *ns* *namespace's* import
ns-name *ns* *namespaces's* name
ns-syms *ns :key*
namespace's symbols

C/C++ API

```
const char* mu_version()
uint64_t mu_t()
uint64_t mu_nil()

bool mu_eof(void*, uint64_t)

uint64_t mu_eval(void*, uint64_t)

uint64_t mu_read_stream(void*, uint64_t)
uint64_t mu_read_cstr(void*, const char*)

void mu_write(void*, uint64_t, uint64_t, bool)
void mu_writeln(void*, uint64_t, uint64_t, bool)

bool mu_with_exception(
     void*,
     bool,
     std::function<void(void*)>)

void* mu_env(int, int, int, char*[], int)
```

Exception Classes and Types

:error :arity function/specop arity
:error :bound *symbol* already bound
:error :dup duplicate *symbol* in lambda list
:error :eof eof during *read*
:error :parse reader parse errors
:error :range *fixnum* out of range
:error :size read object size exceeds limits
:error :syntax reader syntax
:error :type type errors
:error :unbound unbound *symbol* reference
:error :undef undefined symbol or specop
:error :value value out of range

Reader Syntax

`;`	comment to end of line
`#|...|#`	block comment
`(...)`	constant list
`()`	empty list, prints as `:nil`
`'`	quoted form
`"..."`	string
`#x`	hexadecimal *fixnum*
`#d`	decimal *fixnum*
`#\`	character
`#(:type ...)`	vector
`#:symbol`	uninterned *symbol*
`\`	single escape in strings
`"'`,;`	terminating macro char
`#`	non-terminating macro char
`!$%&*+-.`	symbol constituent:
`<>=?@[]|`	
`:^_{}~/`	
`A..Za..z`	
`0..9`	
`backspace`	
`rubout`	
`linefeed`	whitespace:
`newline`	
`page`	
`return`	
`space`	
`tab`	