Mu Namespace

dyad mu version 0.0.11

Type keywords

T	type superclass
`:t :nil`	*bool*
`:char`	*char*
`:cons`	*cons*
`:fixnum`	61 bit signed integer*, fix*
`:float`	32 bit IEEE *float*
`:func`	function*, fn*
`:ns`	*symbol* bindings
`:stream`	*stream:* file, string
`:struct`	struct *vector*
`:symbol`	LISP-1 binding, *symbol*
`:vector`	*vector,* `:t :byte :char :fixnum :float`

Heap

hp-info	*vector,* of type allocations `:type :total :alloc :in-use`

Frame

fr-get *fn*	*struct,* copy frame binding
fr-pop *fn*	*fn,* pop frame binding
fr-push *struct*	*struct,* push frame binding
:fr-ref *fix fix'*	*T,* ref frame variable

Reader/Printer

read *stream bool T*	*T,* read stream object
write *T bool stream*	*T,* write escaped object

Structs

struct *type list*	*struct,* from list
st-type *vector*	*keyword,* struct type
st-vec *vector fix*	*vector,* of members

Symbols

boundp *symbol*	*bool,* is *symbol* bound?
keyp *symbol*	*bool, keyword* predicate
keyword *string*	*symbol, keyword* from *string*
symbol *string*	*symbol,* uninterned *symbol*
sy-ns *symbol*	*symbol ns* binding
sy-name *symbol*	*symbol* name binding
sy-val *symbol*	*symbol* value binding

Special Forms

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

Core

coerce *T :type*	*keyword, coerce* to type
eval *T*	*T,* evaluate form
eq *T T'*	*bool,* are *T* and *T'* identical?
type-of *T*	*keyword*
apply *fn list*	*T,* apply *function* to arg *list*
compile *T*	*T,* library form compiler
with-ex *fn fn'*	*T,* catch exception
raise *:condition T*	*struct,* raise exception
tag-of *T*	*fixnum,* of object tag
view *T*	*struct,* vector of object
fix *fn T*	*T,* fixpoint of *function*
:if *T fn fn'*	*T,* **:if** implementation
***:context**	*cons,* active frame list
***:gc**	*bool,* garbage collection

Fixnums

fx-mul *fix fix'*	*fix,* product of *fix* and *fix'*
fx-add *fix fix'*	*fix,* sum of *fix* and *fix'*
fx-sub *fix fix'*	*fix,* difference of *fix* and *fix'*
fx-lt *fix fix'*	*fix,* is *fix* less than *fix'?*
fx-div *fix fix*	*fix, fix* divided by *fix'*
logand *fix fix'*	*fix,* bitwise *and* of *fix* and *fix'*
logor fix fix'	*fix,* bitwise *or* of fix and fix'

Floats

fl-mul *float float'*	product of *float* and *float'*
fl-add *float float'*	sum of *float* and *float'*
fl-sub *float float'*	difference of *float* and *float'*
fl-lt *float float'*	is *float* less than *float'?*
fl-div *float float'*	*float* divided by *float'*

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*

Vectors

vector *type list*	typed vector from list
sv-len *vector*	length of *vector*
sv-ref *vector fix*	*nth* element
sv-type *vector*	type of *vector*

Namespaces

make-ns *string ns*	make *namespace*
map-ns *string*	map *string* to namespace
intern *ns* `scope` *string value*	intern bound symbol `scope :intern :extern`
ns-map *ns string*	map *string* to *symbol*
ns-imp *ns*	*namespace's* import
ns-name *ns*	*namespace's* name
ns-int *ns*	*namespace's* interns
ns-ext *ns*	*namespace's* externs

Streams

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

open `type dir` *string*	open *stream* from
	`type :file |:string`
	`dir :input|:output`

close *stream*	close *stream*
openp *stream*	is *stream* open?
eof *stream*	is *stream* at end of file?
get-str *stream*	get *vector* from *stream*
rd-byte *stream bool T*	read *byte* from *stream*
un-byte *byte stream*	push *byte* onto *stream*
wr-byte *byte stream*	write *byte* to *stream*
rd-char *stream bool T*	read *char* from *stream*
un-char *char stream*	push *char* onto *stream*
wr-char *char stream*	write *char* to *stream*

Condition Keywords

`:arity`	**`:eof`**	**`:open`**	**`:read`**
`:write`	**`:error`**	**`:syntax`**	**`:type`**
`:unbound`	**`:div0`**	**`:range`**	**`:stream`**

Rust API

```
use crate::mu::core::mu::{
    Exception,
    Extern,
    Mu,
    MuCondition,
    Tag
},

<Mu as Extern>::new(config: String) -> Mu
```

`config:` comma-separated list of **`name:value`** pairs

```
heap:npages
gc:on|off

&'static str <Mu as Extern>::VERSION

pub trait Export for Mu {
  fn nil() -> Tag

  fn eq(tag: Tag, tag1: Tag) -> bool

  fn apply(&self, func: Tag, args) →
             Exception::Result<Tag>

  fn compile(&self, expr: Tag) ->
             Exception::Result<Tag>

  fn eof(&self, stream: Tag) ->
         Exception::Result<Tag>

  fn eval(&self, expr: Tag) ->
          Exception::Result<Tag>

  fn read_stream(&self, stream: Tag,
                 eof: Tag,
                 eof_value: Tag) ->
                 Exception::Result<Tag>

  fn read_string(&self, expr: String) ->
                 Exception::Result<Tag>

  fn write(&self, expr: Tag,
                  escape: bool,
                  stream: Tag) ->
           Exception::Result<()>

  fn write_string(&self, string: String,
                          stream: Tag) ->
                  Exception::Result<()>
}
```

Reader Syntax

`;`	comment to end of line
`#|...|#`	block comment
`(…)`	constant list
`()`	empty list, prints as **`:nil`**
`` ` ``	quoted form
“…”	string/char vector
`#x`	hexadecimal *fixnum*
`#\`	character
`#(:`*`vector-type`* …)	*vector*
`#s(:struct-`*`type`* …)	*struct*
`#:symbol`	uninterned *symbol*
`\`	single escape in strings
`` "`,; ``	terminating macro char
`#`	non-terminating macro char

`!$%&*+-.`
`<>=?@[]|`
`:^_{}~/`
`A..Za..z`
`0..9`
`backspace`
`rubout`
	symbol constituent:

`0x09 tab`
`0x0a linefeed`
`0x0c page`
`0x0d return`
`0x20 space`
	whitespace:

Runtime

```
runtime: 0.0.10: [-h?psvcedlq] [file...]
  ?: usage message
  h: usage message
  c: [name:value,…]
  d: enable debugging
  e: eval [form] and print result
  l: load [path]
  p: pipe mode
  q: eval [form] quietly
  s: script mode
  v: print version and exit
```