

# Mu Namespace

dyad mu version 0.0.6

## Type keywords

<i>T</i>	type superclass
<b>:t :nil</b>	<i>boolean</i>
<b>:char</b>	<i>char</i>
<b>:cons</b>	<i>cons</i>
<b>:fixnum</b>	61 bit signed <i>integer</i>
<b>:float</b>	32 bit IEEE <i>float</i>
<b>:func</b>	<i>function</i>
<b>:ns</b>	<i>symbol</i> bindings
<b>:stream</b>	file, string, socket
<b>:symbol</b>	LISP-1 binding
<b>:vector</b>	<b>:t :byte :char</b> <b>:fixnum :float</b>

## Heap

<b>hp-info</b>	heap values <i>alist</i>
<b>hp-type</b>	<i>type of</i> type occupancy: <i>type</i> : type keyword <i>of</i> : <b>:alloc</b> <b>:in-use</b> <b>:free</b> <b>:size</b>

## Frame

<b>*fr-get func</b>	get frame binding
<b>*fr-setv func fix' fix'</b>	set <i>nth</i> frame binding
<b>*fr-pop func</b>	pop frame binding
<b>*fr-push list</b>	push frame binding
<b>:fr-ref fix fix'</b>	ref frame variable

## Symbols

<b>boundp symbol</b>	<i>symbol</i> bound?
<b>keyp symbol</b>	<i>keyword</i> predicate
<b>keyword string</b>	<i>keyword</i> from <i>string</i>
<b>symbol string</b>	uninterned <i>symbol</i>
<b>sy-ns symbol</b>	<i>symbol ns</i> binding
<b>sy-name symbol</b>	<i>symbol</i> name binding
<b>sy-val symbol</b>	<i>symbol</i> value binding

## Special Forms

<b>:lambda list . body</b>	anonymous <i>function</i>
<b>:quote T</b>	<i>quote</i> form
<b>:if T T' T''</b>	conditional

## Core

<b>coerce T :type</b>	<i>coerce</i> to type keyword
<b>eval T</b>	evaluate form
<b>eq T T'</b>	are <i>T</i> and <i>T'</i> identical?
<b>type-of T</b>	type <i>keyword</i>
<b>apply fn list</b>	apply <i>function</i> to arg <i>list</i>
<b>compile T</b>	library form compiler
<b>with-ex fn fn'</b>	catch exception
<b>raise keyword T</b>	raise exception
<b>tag-of T</b>	object tag to <i>fixnum</i>
<b>*gc</b>	garbage collection
<b>view T</b>	view vector of object
<b>fix fn T</b>	fixpoint <i>function</i>
<b>*fix* fn list</b>	fixpoint <i>function</i>
<b>:if T fn fn'</b>	<b>:if</b> implementation

## Reader/Printer

<b>read stream bool T</b>	read object from stream
<b>write T bool stream</b>	print with escapes

## Fixnums

<b>fx-mul fix fix'</b>	product of <i>fix</i> and <i>fix'</i>
<b>fx-add fix fix'</b>	sum of <i>fix</i> and <i>fix'</i>
<b>fx-sub fix fix'</b>	difference of <i>fix</i> and <i>fix'</i>
<b>fx-lt fix fix'</b>	is <i>fix</i> less than <i>fix'</i> ?
<b>fx-div fix fix'</b>	<i>fix</i> divided by <i>fix'</i>
<b>logand fix fix'</b>	bitwise <i>and</i> of <i>fix</i> and <i>fix'</i>
<b>logor fix fix'</b>	bitwise <i>or</i> <i>fix</i> and <i>fix'</i>

## Floats

<b>fl-mul float float'</b>	product of <i>float</i> and <i>float'</i>
<b>fl-add float float'</b>	sum of <i>float</i> and <i>float'</i>
<b>fl-sub float float'</b>	difference of <i>float</i> and <i>float'</i>
<b>fl-lt float float'</b>	is <i>float</i> less than <i>float'</i> ?
<b>fl-div float float'</b>	<i>float</i> divided by <i>float'</i>

## Lists

<b>car list</b>	head of <i>list</i>
<b>cdr list</b>	tail of <i>list</i>
<b>cons T T'</b>	<i>cons</i> from <i>T</i> and <i>T'</i>
<b>length list</b>	length of <i>list</i>
<b>nth fix list</b>	<i>nth car</i> of <i>list</i>
<b>nthcdr fix list</b>	<i>nth cdr</i> of <i>list</i>

## Vectors

<b>vector type list</b>	specialized vector from <i>list</i>
<b>sv-len vector</b>	<i>fixnum</i> length of <i>vector</i>
<b>sv-ref vector fix</b>	<i>nth</i> element
<b>sv-type vector</b>	type of <i>vector</i> elements

## Condition Keywords

<b>:arity</b>	<b>:eof</b>
<b>:open</b>	<b>:read</b>
<b>:write</b>	<b>:error</b>
<b>:syntax</b>	<b>:type</b>
<b>:unbound</b>	<b>:div0</b>
<b>:range</b>	<b>:stream</b>

## 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*

## 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

## 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  
#:symbol uninterned *symbol*

\ single escape in strings

"` , ;  
# terminating macro char  
non-terminating macro char

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

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

## mu-runtime

mu-runtime: 0.0.6: [-h?psvcelq] [file...]  
?: usage message  
h: usage message  
c: [name:value,...]  
e: eval [form] and print result  
l: load [path]  
p: pipe mode  
q: eval [form] quietly  
s: script mode  
v: print version and exit