

# Mu Namespace

dyad mu version 0.0.7

## Type keywords

<b>T</b>	type superclass
<b>:t :nil</b>	boolean
<b>:char</b>	char
<b>:cons</b>	cons
<b>:fixnum</b>	61 bit signed integer
<b>:float</b>	32 bit IEEE float
<b>:func</b>	function
<b>:ns</b>	symbol 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 alist
<b>hp-type</b>	type of type occupancy: type: type keyword of: <b>:alloc</b> <b>:in-use</b> <b>:free</b> <b>:size</b>

## Frame

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

## Symbols

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

## Special Forms

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

## Core

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

## Reader/Printer

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

## Fixnums

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

## Floats

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

## Lists

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

## Vectors

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