

Mu Runtime Reference

version 0.2.15

type keywords and aliases

<i>supertype</i>	<i>T</i>
<i>bool</i>	<i>()</i> , <i>:nil</i> are false, otherwise true
<i>condition</i>	<i>keyword</i> , see exceptions
<i>list</i>	<i>:cons</i> or <i>()</i> , <i>:nil</i>
<i>ns</i>	<i>#\$(:ns #(:t fixnum symbol))</i>
<i>ns-designator</i>	<i>ns</i> , <i>:nil</i> , <i>:unqual</i>
<i>:null</i>	<i>()</i> , <i>:nil</i>
<i>:char</i>	<i>char</i>
<i>:cons</i>	<i>cons</i> , <i>list</i>
<i>:fixnum</i>	<i>fixnum</i> , <i>fix</i>
<i>:float</i>	<i>float</i> , <i>fl</i>
<i>:func</i>	<i>function</i> , <i>fn</i>
<i>:keyword</i>	<i>keyword</i> , <i>key</i>
<i>:stream</i>	<i>stream</i>
<i>:struct</i>	<i>struct</i>
<i>:symbol</i>	<i>symbol</i> , <i>sym</i>
<i>:vector</i>	<i>vector</i> , <i>string</i> , <i>str</i>
	<i>:bit</i> <i>:char</i> <i>:t</i>
	<i>:byte</i> <i>:fixnum</i> <i>:float</i>

core

<i>apply fn list</i>	<i>T</i>	apply <i>fn</i> to <i>list</i>
<i>compile form</i>	<i>T</i>	<i>mu</i> form compiler
<i>eq T T'</i>	<i>bool</i>	<i>T</i> and <i>T'</i> identical?
<i>eval form</i>	<i>T</i>	evaluate <i>form</i>
<i>type-of T</i>	<i>key</i>	type keyword
<i>view for</i>	<i>vector</i>	vector of object
<i>fix fn T</i>	<i>T</i>	fixpoint of <i>fn</i>
<i>gc</i>	<i>bool</i>	garbage collection
<i>repr T</i>	<i>vector</i>	tag representation
<i>unrepr vector</i>	<i>T</i>	tag representation

tag vector is an 8 element :byte vector
of little-endian argument tag bits.

special forms

<i>:lambda list . list'</i>	<i>function</i>	anonymous <i>fn</i>
<i>:alambda list . list'</i>	<i>function</i>	anonymous <i>fn</i>
<i>:quote T</i>	<i>list</i>	quoted form
<i>:if T T' T''</i>	<i>T</i>	conditional

frames			vectors		
		frame binding: <i>(fn . #(:t ...))</i>			
%frame-stack	<i>list</i>	active frames	make-vector	<i>key list</i>	specialized vector from list
%frame-pop fn	<i>frame</i>	pop function's top frame binding	vector-length	<i>vector</i>	length of vector
%frame-push frame	<i>cons</i>	push frame	vector-type	<i>vector</i>	type of vector
%frame-ref fn fix	<i>T</i>	function, offset	sref	<i>vector fix</i>	<i>n</i> th element
symbols			namespaces		
boundp	<i>sym</i>	<i>bool</i>	runtime namespaces: <i>mu</i> (static), keyword		
make-symbol	<i>string</i>	<i>sym</i>	make-namespace	<i>str</i>	make namespace
symbol-namespace	<i>sym</i>	<i>ns-designator</i>	namespace-name	<i>ns</i>	namespace name
symbol-name	<i>symbol</i>	<i>ns</i> -designator	intern	<i>ns str value</i>	intern symbol in non-static namespace
symbol-value	<i>symbol</i>	<i>string</i>	find-namespace	<i>str</i>	map string to namespace
		<i>T</i>	find	<i>ns string</i>	map string to symbol
fixnums			structs		
add fix fix'		<i>fixnum</i>	make-struct	<i>key list</i>	type key from list
ash fix fix'		<i>fixnum</i>	struct-type	<i>struct</i>	struct type key
div fix fix'		<i>fixnum</i>	struct-vec	<i>vector</i>	of struct members
less-than fix fix'		<i>bool</i>			
logand fix fix'		<i>fixnum</i>			
lognot fix		<i>fixnum</i>			
logor fix fix'		<i>fixnum</i>			
mul fix fix'		<i>fixnum</i>			
sub fix fix'		<i>fixnum</i>			
floats			streams		
fadd fl fl'		<i>float</i>	*standard-input*	<i>stream</i>	std input stream
fdiv fl fl'		<i>float</i>	*standard-output*	<i>stream</i>	std out stream
fless-than fl fl'		<i>bool</i>	*error-output*	<i>stream</i>	std error stream
fmul fl fl'		<i>float</i>	open	<i>type dir str bool</i>	open stream, raise error if bool
fsub fl fl'		<i>float</i>		<i>type dir :file :input</i>	:file :string :output :bidir
conses/lists					
append list		<i>list</i>			
car list		<i>T</i>			
cdr list		<i>T</i>			
cons T T'		<i>cons</i>			
length list		<i>fixnum</i>			
nth fix list		<i>T</i>			
nthcdr fix list		<i>T</i>			

exceptions	environment config	Reader Syntax																																																																								
<p>with-exception <i>fn fn'</i> <i>T</i> catch exception</p> <pre>fn - (:lambda (obj cond src) . body) fn'-(:lambda () . body)</pre> <p>raise <i>T T' keyword</i> raise exception on <i>T</i> from source designator <i>T'</i> with <i>keyword</i> condition</p> <pre>:arity :div0 :eof :error :except :future :ns :open :over :quasi :range :read :exit :signal :stream :syntax :syscall :type :unbound :under :write :storage :user</pre>	<p>JSON config format:</p> <pre>{ "pages": <i>N</i>, "gc-mode": "none" "auto", }</pre> <p>Mu library API</p> <pre>[dependencies] mu = { git = "https://github.com/Software-Knife-and-Tool/mu.git", branch = "main" } use mu::{ Mu, Env, Config }; use mu::{ Result, Tag, Exception, Condition }; impl Mu { fn apply(_: &Env, _: Tag, _: Tag) → Result<Tag> fn compile(_: &Env, _: Tag) → Result<Tag> fn config(_: Option<String>) → Option<Config> fn eq(_: Tag, _: Tag) → bool; fn err_out() → Tag fn eval_str(_: &Env, _: &str) → Result<Tag> fn eval(_: &Env, _: Tag) → Result<Tag> fn exception_string(_: &Env, _: Exception) → String fn load(_: &Env, _: &str) → Result<bool> fn make_env(_: &Config) → Env fn read_str(_: &Env, _: &str) → Result<Tag> fn read(_: &Env, _: Tag, _: bool, _: Tag) → Result<Tag> fn std_in() → Tag fn std_out() → Tag fn version() → &str fn write_str(_: &Env, _: &str, _: Tag) → Result<()> fn write_to_string(_: &Env, _: Tag, _: bool) → String fn write(_: &Env, _: Tag, _: bool, _: Tag) → Result<()> }</pre> <p>API function argument details:</p> <pre> apply &Env function list → Result<Tag> compile &Env T → Result<Tag> config Option<String> → Option<Config> eq T T' → bool err_out → Tag eval_str &Env &str → Result<Tag> eval &Env T → Result<Tag> exception_string &Env Exception → String load &Env &str → Result<bool> env &Config → Env read_str &Env &str → Result<Tag> read &Env stream bool bool → Result<Tag> // bool – raise exception on end of stream std_in → Tag std_out → Tag version → &str write_str &Env &str stream → Result<()> write_to_string &Env T bool bool → String // bool – print escaped write &Env T bool stream → Result<()> // bool – print escaped </pre>	<p>comment to end of line</p> <p>block comment</p> <p>quoted form</p> <p>backquoted form</p> <p>backquoted list (proper lists)</p> <p>eval backquoted form</p> <p>eval-splice backquoted form</p> <p>constant <i>list</i></p> <p>empty <i>list</i>, prints as :nil</p> <p>dotted <i>list</i></p> <p><i>string, char vector</i></p> <p>single escape in strings</p> <p>qualified <i>symbol</i>, where <i>ns</i> and <i>name</i> are <i>symbol constituents</i></p> <p>lexical <i>symbol</i></p> <p>bit vector</p> <p>hexadecimal <i>fixnum</i></p> <p>read-time eval</p> <p><i>char</i></p> <p><i>vector</i></p> <p><i>struct</i></p> <p>uninterned <i>symbol</i></p> <p>terminating macro <i>char</i></p> <p>non-terminating macro <i>char</i></p> <p><i>symbol constituent</i></p> <p>character designators</p>																																																																								
<h2>Features</h2> <pre>[features] default = ["core", "env", "system"]</pre> <table border="1"> <thead> <tr> <th>feature/core</th> <th>core</th> <th>list</th> <th>core state</th> </tr> </thead> <tbody> <tr> <td></td> <td>sleep</td> <td>fixnum</td> <td>microseconds</td> </tr> <tr> <td></td> <td>process-mem-virt</td> <td>fixnum</td> <td>vmem</td> </tr> <tr> <td></td> <td>process-mem-res</td> <td>fixnum</td> <td>reserve</td> </tr> <tr> <td></td> <td>process-time</td> <td>fixnum</td> <td>microseconds</td> </tr> <tr> <td></td> <td>time-units-per-sec</td> <td>fixnum</td> <td></td> </tr> <tr> <td></td> <td>ns-symbols ns :nil</td> <td>list</td> <td>symbol list</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>feature/env</th> <th>env</th> <th>list</th> <th>env state</th> </tr> </thead> <tbody> <tr> <td></td> <td>heap-info</td> <td>0</td> <td>heap info to stdout</td> </tr> <tr> <td></td> <td>heap-room key</td> <td>vector #(:t size total free ...)</td> <td>allocations</td> </tr> <tr> <td></td> <td>heap-size key</td> <td>fixnum</td> <td>type size</td> </tr> <tr> <td></td> <td>cache-room</td> <td>vector #(:t size total ...)</td> <td>allocations</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>feature/system</th> <th>uname</th> <th>:t</th> <th>system info</th> </tr> </thead> <tbody> <tr> <td></td> <td>shell string list</td> <td>fixnum</td> <td>shell command</td> </tr> <tr> <td></td> <td>exit fixnum</td> <td></td> <td>doesn't return</td> </tr> <tr> <td></td> <td>sysinfo</td> <td>:t</td> <td>not on macOS</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>feature/instrument</th> <th>instrument-control</th> <th>key</th> <th>:on :off :get</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td>key vec</td> <td></td> </tr> </tbody> </table>	feature/core	core	list	core state		sleep	fixnum	microseconds		process-mem-virt	fixnum	vmem		process-mem-res	fixnum	reserve		process-time	fixnum	microseconds		time-units-per-sec	fixnum			ns-symbols ns :nil	list	symbol list	feature/env	env	list	env state		heap-info	0	heap info to stdout		heap-room key	vector #(:t size total free ...)	allocations		heap-size key	fixnum	type size		cache-room	vector #(:t size total ...)	allocations	feature/system	uname	:t	system info		shell string list	fixnum	shell command		exit fixnum		doesn't return		sysinfo	:t	not on macOS	feature/instrument	instrument-control	key	:on :off :get			key vec		<pre>:</pre> <pre># . . . #</pre> <pre>'form</pre> <pre>'form</pre> <pre>'(..)</pre> <pre>.form</pre> <pre>,@form</pre> <pre>(...)</pre> <pre>()</pre> <pre>(... . .)</pre> <pre>"..."</pre> <pre> </pre> <pre>ns:name</pre> <pre>name</pre> <pre>#*</pre> <pre>#X</pre> <pre>#.</pre> <pre>#\</pre> <pre>#(:type ...)</pre> <pre>#\$(:(type ...))</pre> <pre>#::...</pre> <pre>"`,";</pre> <pre>#</pre> <pre>!\$%&*+-.</pre> <pre><>=?@[[]]</pre> <pre>:^_{}~/</pre> <pre>A..Za..z</pre> <pre>0..9</pre> <pre>0x09 #\tab</pre> <pre>0x0a #\linefeed</pre> <pre>0x0c #'page</pre> <pre>0x0d #'return</pre> <pre>0x20 #\space</pre>	
feature/core	core	list	core state																																																																							
	sleep	fixnum	microseconds																																																																							
	process-mem-virt	fixnum	vmem																																																																							
	process-mem-res	fixnum	reserve																																																																							
	process-time	fixnum	microseconds																																																																							
	time-units-per-sec	fixnum																																																																								
	ns-symbols ns :nil	list	symbol list																																																																							
feature/env	env	list	env state																																																																							
	heap-info	0	heap info to stdout																																																																							
	heap-room key	vector #(:t size total free ...)	allocations																																																																							
	heap-size key	fixnum	type size																																																																							
	cache-room	vector #(:t size total ...)	allocations																																																																							
feature/system	uname	:t	system info																																																																							
	shell string list	fixnum	shell command																																																																							
	exit fixnum		doesn't return																																																																							
	sysinfo	:t	not on macOS																																																																							
feature/instrument	instrument-control	key	:on :off :get																																																																							
		key vec																																																																								