# *Mu Runtime Reference*

**version *0.2.14***

## *type keywords and aliases*

| | | |
|---|---|---|
| *supertype* | *T* | |
| *bool* | `()`, `:nil` are false, otherwise true | |
| *condition* | *keyword,* see **exceptions** | |
| *list* | `:cons` or `()`, `:nil` | |
| *ns* | `#s(:ns #(:t` *fixnum symbol*`))` | |
| *ns -designator* | *ns*, `:nil`, `:unqual` | |
| `:null` | `()`, `:nil` | |
| `:char` | *char* | 8 bit ASCII |
| `:cons` | *cons, list* | list, cons, dotted pair |
| `:fixnum` | *fixnum, fix* | 56 bit signed integer |
| `:float` | *float, fl* | 32 bit IEEE float |
| `:func` | *function, fn* | function |
| `:keyword` | *keyword, key* | symbol |
| `:stream` | *stream* | file or string type |
| `:struct` | *struct* | see **structs** |
| `:symbol` | *symbol, sym* | LISP-1 symbol |
| `:vector` | *vector, string, str* | typed vector |

`:bit :char :t`
`:byte :fixnum :float`

## *core*

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

## *special forms*

| | | |
|---|---|---|
| `:lambda` *list . list'* | *function* | anonymous *fn* |
| `:alambda` *list . list'* | *function* | anonymous *fn* |
| `:quote` *T* | *list* | quoted form |
| `:if` *T T' T''* | *T* | *conditional* |

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

## *frames*

*frame* binding: `(`*fn* `. #(:t …))`

| | | |
|---|---|---|
| ***%frame-stack*** | *list* | active *frames* |
| ***%frame-pop*** *fn* | *frame* | pop *function's* top frame binding |
| ***%frame-push*** *frame* | *cons* | push frame |
| ***%frame-ref*** *fn fix* | *T* | *function*, offset |

## *symbols*

| | | |
|---|---|---|
| ***boundp*** *sym* | *bool* | is *symbol* bound? |
| ***make-symbol*** *string* | *sym* | uninterned *symbol* |
| ***symbol-namespace*** *sym* | *ns-designator* | |
| | *namespace designator* | |
| ***symbol-name*** *symbol* | *string* | *name binding* |
| ***symbol-value*** *symbol* | *T* | value binding |

## *fixnums*

| | | |
|---|---|---|
| ***add*** *fix fix'* | *fixnum* | sum |
| ***ash*** *fix fix'* | *fixnum* | arithmetic shift |
| ***div*** *fix fix'* | *fixnum* | quotient |
| ***less-than*** *fix fix'* | *bool* | *fix < fix'?* |
| ***logand*** *fix fix'* | *fixnum* | bitwise and |
| ***lognot*** *fix* | *fixnum* | bitwise complement |
| ***logor*** *fix fix'* | *fixnum* | bitwise or |
| ***mul*** *fix fix'* | *fixnum* | product |
| ***sub*** *fix fix'* | *fixnum* | difference |

## *floats*

| | | |
|---|---|---|
| ***fadd*** *fl fl'* | *float* | sum |
| ***fdiv*** *fl fl'* | *float* | quotient |
| ***fless-than*** *fl fl'* | *bool* | *fl < fl'?* |
| ***fmul*** *fl fl'* | *float* | product |
| ***fsub*** *fl fl'* | *float* | difference |

## *conses/lists*

| | | |
|---|---|---|
| ***append*** *list* | *list* | append lists |
| ***car*** *list* | *T* | head of *list* |
| ***cdr*** *list* | *T* | tail of *list* |
| ***cons*** *T T'* | *cons* | (*T* . *T'*) |
| ***length*** *list* | *fixnum* | length of *list* |
| ***nth*** *fix list* | *T* | *n*th car of *list* |
| ***nthcdr*** *fix list* | *T* | *n*th cdr of *list* |

## *vectors*

| | | |
|---|---|---|
| ***make-vector*** *key list* | *vector* | specialized *vector* from list |
| ***vector-length*** *vector* | *fixnum* | length of *vector* |
| ***vector-type*** *vector* | *key* | type of *vector* |
| ***svref*** *vector fix* | *T* | *n*th element |

## *namespaces*

*runtime namespaces: mu (static), keyword*

| | | |
|---|---|---|
| ***make-namespace*** *str* | *ns* | make *namespace* |
| ***namespace-name*** *ns* | *string* | *namespace* name |
| ***intern*** *ns str value* | *symbol* | intern symbol in non-static *namespace* |
| ***find-namespace*** *str* | *ns* | map *string* to *namespace* |
| ***find*** *ns string* | *symbol* | map *string* to *symbol* |

## *structs*

| | | |
|---|---|---|
| ***make-struct*** *key list* | *struct* | type *key* from *list* |
| ***struct-type*** *struct* | *key* | *struct* type *key* |
| ***struct-vec*** *struct* | *vector* | of *struct* members |

## *streams*

| | | |
|---|---|---|
| ****standard-input**** | *stream* | std input *stream* |
| ****standard-output**** | *stream* | std out *stream* |
| ****error-output**** | *stream* | std error *stream* |
| **open** *`type` `dir` str bool* | *stream* | open *stream,* raise error if *bool* |

```
type    :file  :string
dir     :input :output :bidir
```

| | | |
|---|---|---|
| ***close*** *stream* | *bool* | close *stream* |
| ***openp*** *stream* | *bool* | is *stream* open? |
| ***flush*** *stream* | *bool* | flush *steam* |
| ***get-string*** *stream* | *string* | from *string stream* |
| ***read-byte*** *stream bool T* | *byte* | read *byte* from *stream,* error on eof, *T:* eof-value |
| ***read-char*** *stream bool T* | *char* | read *char* from *stream*, error on `eof`, *T:* `eof`-value |
| ***unread-char*** *char stream* | *char* | push *char* onto *stream* |
| ***write-byte*** *byte stream* | *byte* | write *byte* |
| ***write-char*** *char stream* | *char* | write *char* |
| ***read*** *stream bool T* | *T* | read *stream* |
| ***write*** T *bool stream* | *T* | write with escape |

## exceptions

| | | |
|---|---|---|
| ***with-exception*** *fn fn'* | *T* | catch exception |

*fn* - `(:lambda (`*obj cond src*`) .` *body*`)`
*fn'*- `(:lambda () .` *body*`)`

| | |
|---|---|
| ***raise*** *T keyword* | raise exception on *T* with *keyword* condition |
| ***raise-from*** *T symbol keyword* | raise exception on *T* with *keyword* condition |

```
:arity   :div0    :eof     :error   :except
:future  :ns      :open    :over    :quasi
:range   :read    :exit    :signal  :stream
:syntax  :syscall :type    :unbound :under
:write   :storage :user
```

## Features

```
[features]
default = [ “core”, “env”, "system" ]
```

| | | | |
|---|---|---|---|
| **feature/core** | ***core*** | *list* | core state |
| | ***delay*** | *fixnum* | microseonds |
| | ***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*** | *()* | heap info to stdout |
| | ***heap-room*** *key* `#(:t size total free …)` | *vector* | allocations |
| | ***heap-size*** *key* | *fixnum* | type size |
| | ***cache-room*** `#(:t size total …)` | *vector* | allocations |
| ***feature*/system** | ***uname*** | `:t` | system info |
| | ***shell*** *string list* | *fixnum* | shell command |
| | ***exit*** *fixnum* | | |
| | ***sysinfo*** | `:t` | not on macOS |
| ***feature/instrument*** | ***instrument-control*** *key* | *key* \| *vec* | `:on|:off|:get` |

## environment

```
JSON config format:

{
    “pages”: N,
    “gc-mode”: “none” | “auto”,
}
```

## Mu library API

```
[dependencies]
mu = {
  git = “https://github.com/Software-Knife-and-Tool/mu.git”,
  branch = “main”
}

use mu::{ Condition, Core, Env, Exception,
          Mu, Result, Tag };

impl Mu {
  fn apply(_: &Env, _: Tag, _: Tag) → Result<Tag>
  fn compile(_: &Env, _: Tag) → Result<Tag>
  fn config(_: Option<String>) → Option<Config>
  fn core() → &Core
  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<()>
}
```

## Reader Syntax

| | |
|---|---|
| `;` | comment to end of line |
| `#|...|#` | block comment |
| ‘*form* | quoted form |
| \`*form* | backquoted form |
| \` (…) | backquoted list (proper lists) |
| ,*form* | eval backquoted form |
| ,@*form* | eval-splice backquoted form |
| `(…)` | constant *list* |
| `()` | empty *list*, prints as `:nil` |
| `(… . .)` | dotted *list* |
| “…” | *string, char* vector |
| `\` | single escape in strings |
| *ns:name* | qualified *symbol,* where *ns* and *name* are *symbol constituents* |
| *name* | lexical *symbol* |
| #* | bit vector |
| #x | hexadecimal *fixnum* |
| `#.` | read-time eval |
| `#\` | *char* |
| `#(:type …)` | *vector* |
| `#s(:type …)` | *struct* |
| `#:…` | uninterned *symbol* |
| “ \` , ; | terminating macro char |
| # | non-terminating macro char |
| `!$%&*+-.` `<>=?@[]|` `:^_{}~/` `A..Za..z` `0..9` | *symbol constituent* |
| | character designators |

```
0x09 #\tab
0x0a #\linefeed
0x0c #\page
0x0d #\return
0x20 #\space
```