# libcore *Reference*

***libcore* version *0.0.39***

## *Type Keywords and aliases*

| | |
|---|---|
| *supertype* | *T* |
| *bool* | `()`, `:nil` are false, otherwise true |
| *condition* | *keyword,* see ***Exception*** |
| *list* | *cons* or `()`, `:nil` |
| *frame* | *cons*, see ***Frame*** |

| | | |
|---|---|---|
| `:null` | `()`, `:nil` | |
| `:asyncid` | *async* | async future id |
| `:char` | *char* | |
| `:cons` | *cons* | |
| `: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* | typed vector |
| `:symbol` | *symbol, sym* | LISP-1 symbol |
| `:vector` | *vector, string, str* | `:char :t :byte :fixnum :float` |

## *Heap* *p*

| | | |
|---|---|---|
| **hp-info** | *vector* | heap static information<br>`#(:t type pages pagesize)` |
| **hp-stat** | *vector* | heap allocations<br>`#(:t :type size total free ...)` |
| **hp-size** *T* | *fixnum* | heap occupancy in bytes |

## *Frame* *e*

*frame* binding: `(`*fn* `. #(:t …))`

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

## *Struct* *t*

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

## *Symbol* *l*

| | | |
|---|---|---|
| **boundp** *sym* | *bool* | is *symbol* bound? |
| **keyword** *str* | *key* | *keyword* from *string* |
| **symbol** *str* | *symbol* | uninterned *symbol* |
| **sy-ns** *sym* | *key* | *symbol* namespace |
| **sy-name** *sym* | *string* | *symbol* name binding |
| **sy-val** *sym* | *T* | *symbol* value binding |

## *Special Forms* *s*

| | | |
|---|---|---|
| ***:async** *fn . list* | *async* | create *future* context |
| **:lambda** *list . list'* | *function* | anonymous function |
| **:quote** *form* | *list* | quoted form |
| **:if** *form T T'* | *T* | conditional |

## *Core* *s*

| | | |
|---|---|---|
| **apply** *fn list* | *T* | apply *function* to *list* |
| **eval** *form* | *T* | evaluate *form* |
| **eq** *T T'* | *bool* | *are T* and *T'* identical? |
| **type-of** *T* | *keyword* | |
| ***await** *async* | *T* | return value of async future |
| ***abort** *async* | *T* | abort future |
| **compile** *form* | *T* | *mu* form compiler |
| **view** *form* | *vector* | vector of object |
| **repr** `type` *T* | *T* | tag representation |

`type` `- :t :vector`

if `type` is `:vector`, return 8 byte byte vector of argument tag bits, otherwise convert argument byte vector to tag.

| | | |
|---|---|---|
| **fix** *fn form* | *T* | fixpoint of *function* on *form* |
| **gc** *bool* | *bool* | garbage collection, verbose |

## *Fixnum* *m*

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

## *Float* *t*

| | | |
|---|---|---|
| **fl-mul** *fl fl'* | *float* | product |
| **fl-add** *fl fl'* | *float* | sum |
| **fl-sub** *fl fl'* | *float* | difference |
| **fl-lt** *fl fl'* | *bool* | *fl < fl'?* |
| **fl-div** *fl fl'* | *float* | quotient |

## *Conses/Lists* *s*

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

## *Vector* *s*

| | | |
|---|---|---|
| **vector** *key list* | *vector* | specialized vector from list |
| **sv-len** *vector* | *fixnum* | length of *vector* |
| **sv-ref** *vector fix* | *T* | *n*th element |
| **sv-type** *vector* | *key* | type of *vector* |

## *System*

| | | |
|---|---|---|
| **sys-tm** | *fixnum* | system time in *us* |
| **proc-tm** | *fixnum* | process time in *us* |
| **getpid** | *fixnum* | process id |
| **getcwd** | *string* | `getcwd(2)` |
| **uname** | | *struct* `uname(2)` |
| **spawn** *str list* | *fixnum* | spawn command |
| **sysinfo** | | *struct* `sysinfo(2)` |
| **exit** | *fixnum* | exit shell with fixnum |

## Exception

**with-ex** *fn fn' T* — catch exception
*fn* - `(:lambda (obj cond src) . body)`
*fn'* - `(:lambda () . body)`

**raise** *T keyword* — raise exception with condition

```
:arity  :eof    :open   :read :syscall
:write  :error  :syntax :type
:div0   :stream :range  :except
:ns     :over   :under  :unbound
```

## Stream

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

**open** *type direction string*
*stream* — open *stream*
*type* - `:file :string`
*direction* - `:input :output :bidir`

| | | |
|---|---|---|
| **close** *stream* | *bool* | close *stream* |
| **openp** *stream* | *bool* | is *stream* open? |
| **flush** *stream* | *bool* | flush output steam |
| **get-str** *stream* | *string* | from *string stream* |

**rd-byte** *stream bool T*
*byte* — read *byte* from *stream*, error on eof, *T:* eof value

**rd-char** *stream bool T*
*char* — read *char* from *stream*, error on `eof`, *T:* `eof` value

**un-char** *char stream*
*char* — push *char* onto *stream*

**wr-byte** *byte stream*
*byte* — write *byte* to *stream*

**wr-char** *char stream*
*char* — write *char* to *stream*

## Namespace

**make-ns** *key* *key* — make namespace
**ns-map** *list* — list of mapped namespaces
**untern** *key string*
*symbol* — intern unbound symbol
**intern** *key string value*
*symbol* — intern bound symbol
**ns-find** *key string*
*symbol* — map *string* to *symbol*
**ns-syms** `type` *key*
*T* — namespace's *symbols*
`type` - `:list :vector`

## Reader/Printer

**read** *stream bool T*
*T* — read stream object

**write** *T bool stream*
*T* — write escaped object

## libcore API

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

use libcore::{Condition, Config, Exception, Mu, Result, Tag}

config string format: "npages:N,gcmode:GCMODE"
GCMODE - { none, auto, demand }

impl Mu {
  const Mu::VERSION: &str

  fn config(config: String) -> Option<Config>;
  fn new(config: &Config)-> Mu;

  fn apply(&self, func: Tag, args: Tag)-> Result;
  fn compile(&self, form: Tag) -> Result;
  fn eq(&self, func: Tag, args: Tag) -> bool;
  fn exception_string(&self, ex: Exception) -> String;
  fn eval(&self, expr: Tag) -> Result;
  fn eval_str(&self, expr: &str) -> Result;
  fn load(&self, file_path: &str) -> Result;
  fn load_image(&self, file_path: &str) -> Result;
  fn read(&self, stream: Tag, eofp: bool, eof: Tag) -> Result;
  fn read_str(&self, str: &str) -> Result;
  fn err_out(&self) -> Tag
  fn save_and_exit(&self, file_path: &str) → Result;
  fn std_in(&self) -> Tag
  fn std_out(&self) -> Tag
  fn write(&self, expr: Tag, esc: bool, stream: Tag) -> Result
  fn write_str(&self, str: &str, stream: Tag) -> Result;
  fn write_to_string(&self, stream: Tag) -> Result;
}
```

## Reader Syntax

| | |
|---|---|
| ; | comment to end of line |
| `#\|...\|#` | block comment |
| '*form* | quoted form |
| `*form* | backquoted form |
| `(...) | backquoted list (proper lists only) |
| ,*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 |
| #x | hexadecimal *fixnum* |
| #\c | *char* |
| #(:type ...) | *vector* |
| #s(:type ...) | *struct* |
| #:symbol | uninterned *symbol* |
| "`,; | terminating macro char |
| # | non-terminating macro char |

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

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

## Runtime

```
mu-sys: x.y.z: [-h?pvcelq] [file…]

?: usage message
h: usage message
c: [name:value,…]
e: eval [form] and print result
l: load [path]
p: pipe mode (no repl)
q: eval [form] quietly
v: print version and exit
```