Wasp Lisp Articles

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
	5 January 2011		CD

Contents

1 Building Wasp Lisp

The source for the Wasp Lisp VM is held in a *github* source code repository. You can get the source with the command:

```
$ git clone git://github.com/swdunlop/WaspVM.git
```

This retrieves the source into a directory called *WaspVM*. Build by running *make*:

```
$ cd WaspVM
~WaspVM$ make
```

Once built you can run the REPL from the build directory with:

```
~WaspVM$ make repl
```

Another way to run the REPL is to run the *wasp* program from the *mod* subdirectory. I recommend using a program like *rlwrap* to give command line history at the REPL:

```
~WaspVM$ cd mod
~WaspVM/mod$ rlwrap ../wasp
>> (print  "hello")
hello:: null
>>
```

2 Compiling Wasp Lisp Programs

Wasp Lisp programs can be compiled into an executable. This is done by compiling the Wasp Lisp code into a bytecode format. The bytecode is then appended to a machine language *stub* file which is specific to the platform the executable will be run on.

The stub is generated as part of the normal *make* process and is in the *stubs* subdirectory. You can copy stub files produced from builds on other platforms into the *stubs* subdirectory to produce executables than run on platforms other than the one you are using.

The command to do all this is *waspc*. It takes up to three options. They are:

-exe <dest-path>	The executable that is created when compiling
-platform <platform>	Specifies which OS and architecture to compile for
-stub <stub-path>	Like -platform but specifies the actual architecture and platform specific stub file to use.

An example of generating a *hello world* program:

```
~$ cat >test.ms
(define (main)
  (print
~$ waspc -exe hello test.ms
BUILD: test
BUILD: core/macro
BUILD: core/config
BUILD: site/config
BUILD: core/file
BUILD: core/module
BUILD: core/io
BUILD: core/macro
BUILD: core/config
BUILD: site/config
BUILD: core/file
BUILD: core/module
```

```
BUILD: core/io
BUILD: test
~$ chmod +x hello
~$ ./hello
hello world!
```

3 Function Reference

A function reference.

3.1 Modules

3.1.1 lib/eval

```
IMPORTS: lib/compile, lib/optimize
EXPORTS: (eval expr)
         (exec block)
         (load path)
         (load-ms path)
```

3.1.2 lib/clue

```
IMPORTS: lib/object
EXPORTS: (clue-db->list db)
         (clue-db-records clue-db)
         (clue-record->list record)
         (clue-records-with-key db key)
         (clue-records-with-parameter db key value)
         (clue-union set0 sets ...)
         (drop-clue records)
         (find-clue-records db parameters ...)
         (get-clue-record-parameters record)
         (get-clue-record-value record key)
         (list->clue-db data)
         (list->clue-record db data)
         (new-clue-db)
         (new-clue-record db parameters)
         (set-clue-record-parameter record key value)
         (set-clue-record-parameters record parameters)
```

3.1.3 lib/trace

```
IMPORTS: lib/filter
EXPORTS:
MODULE:  ./lib/object
IMPORTS: lib/iterate
EXPORTS: (class-fields (<tag> class))
         (make-class class-name super-class fields)
         (make-class-constructor class-name class arg-names)
         (make-field-accessor class field-name)
         (make-field-modifier class field-name)
         object
```

3.1.4 lib/compile

```
IMPORTS: lib/iterate, lib/module, lib/record
EXPORTS: (compile program)
         (make-branch-symbol root)
         (make-symbol items ...)
         (reset-branch-index)
         symbol-starts-withq
         (waspvm-special-forms)
```

3.1.5 lib/format-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.6 lib/http-url

```
IMPORTS: lib/url
EXPORTS: <http-url>
         (http-url-arg url arg)
         (http-url-args http-url)
         (http-url-frag http-url)
         (http-url-host http-url)
         (http-url-path http-url)
         (http-url-portno http-url)
         (http-url-user http-url)
         (http-url? value)
         (make-http-url user host portno path args frag)
         (url-auth (<http-url> url))
         (url-frag (<http-url> url))
         (url-path (<http-url> url))
         (url-query (<http-url> url))
         (url-scheme (<http-url> url))
```

3.1.7 lib/http-server

```
IMPORTS: lib/http-url, lib/tcp-server
EXPORTS: <http-request>
         (http-request-arg http-request key)
         (http-request-body http-request)
         (http-request-cookie http-request key)
         (http-request-cookies http-request)
         (http-request-header http-request key)
         (http-request-headers http-request)
         (http-request-input http-request)
         (http-request-method http-request)
         (http-request-output http-request)
         (http-request-url http-request)
         (http-request-version http-request)
         (http-request? value)
         (http-response-cookie cookie)
         (http-response-cookies cookies ...)
         (http-service http-responder)
         (read-http-request input)
         (set-http-request-input! http-request input)
         (spawn-http-server portno http-responder)
         (write-http-response port code reason headers body)
```

3.1.8 lib/package-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.9 lib/module

```
IMPORTS:
EXPORTS: (code-dependencies source)
         (code-exports module)
         (code-imports module)
         (core-dependencies)
```

3.1.10 lib/line-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.11 lib/copy-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.12 lib/scud

```
IMPORTS: lib/env, lib/record
EXPORTS: (scud-print data ...)
         (set-scud-bg color)
         (set-scud-colors bg fg)
         (set-scud-fg color)
```

3.1.13 lib/eval

```
IMPORTS: lib/compile, lib/optimize
EXPORTS: (eval expr)
         (exec block)
         (load path)
         (load-ms path)
```

3.1.14 lib/spawn-connection

```
IMPORTS:
EXPORTS: (spawn-connection func rest ...)
```

3.1.15 lib/fuzz-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.16 lib/test

```
IMPORTS: lib/iterate
EXPORTS: (disable-spot-tests)
         (do-r expr fn)
         (do-s expr fn)
         (do-t expr fn)
         (enable-spot-tests)
         (main ignored ...)
         (spot-report)
         spot-s
         spot-v
```

3.1.17 lib/socks-client

```
IMPORTS:
EXPORTS: (format-socks4-request auth proto addr portno)
         (parse-socks4-response str)
         (tcp-socks4-connect auth socks-addr socks-portno dest-addr dest-portno)
```

3.1.18 lib/crypto-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.19 lib/build

```
IMPORTS: lib/compile, lib/eval, lib/optimize, lib/waspc
EXPORTS: (build-dependencies source)
         (build-exe platform main)
         (build-exe-with-stub stub main)
         (build-module module)
         (build-modules modules ...)
         (read-platform-stub platform)
```

3.1.20 lib/url

```
IMPORTS: lib/object
EXPORTS: (make-url scheme auth path query frag)
         (register-url-parser key parser)
         (string->url str (<string> scheme))
         <url>
         (url-auth url)
         (url-frag url)
         url-parsers
         (url-path url)
         (url-query url)
         url-regex
         (url-scheme url)
         (url->string url)
         (url? value)
```

3.1.21 lib/shuffle

```
IMPORTS:
EXPORTS: (shuffle x)
         (shuffle/list l)
         (shuffle/vector v)
```

3.1.22 lib/with-io

```
IMPORTS:
EXPORTS: (do-with-input new-input func)
         (do-with-io new-input new-output func)
         (do-with-output new-output func)
```

3.1.23 lib/tcp-server

```
IMPORTS: lib/with-io
EXPORTS: (spawn-tcp-server portno fn rest ...)
```

3.1.24 lib/filter

```
IMPORTS: lib/iterate
EXPORTS: (error-on-fail message)
         (input-chain recv filters ...)
         (output-chain xmit filters ...)
```

3.1.25 lib/s-filter

```
IMPORTS: lib/filter, lib/iterate, lib/options
EXPORTS:
```

3.1.26 lib/record

```
IMPORTS: lib/object
EXPORTS:
```

3.1.27 lib/http-client

```
IMPORTS: lib/http-url, lib/object, lib/url
EXPORTS: (http-get url)
         (http-post url mime body)
         (http-response-body http-response)
         (http-response-code http-response)
         (http-response-input http-response)
         (http-response-message http-response)
         (http-response-output http-response)
         (http-response? value)
         (make-http-response code message headers body input output)
         (read-http-response channel)
         (send-http-get channel url)
         (send-http-post channel url mime body)
```

3.1.28 lib/buffer-input

```
IMPORTS: lib/catch
EXPORTS: (buffer-input source timeout debris)
         (buffer-input/eoc rest ...)
         (error-on-close fn)
```

3.1.29 lib/collate-filter

```
IMPORTS: lib/buffer-input, lib/filter
EXPORTS:
```

3.1.30 lib/catch

```
IMPORTS:
EXPORTS:
```

3.1.31 lib/env

```
IMPORTS: lib/iterate
EXPORTS: env
         (env-is name val)
         (get-env name)
         (has-env name)
         in-darwin
         in-macosx
         in-posix
         in-win32
         in-winnt
         in-x11
         (locate-cmd name)
```

3.1.32 lib/socks-server

```
IMPORTS: lib/patch, lib/tcp-server, lib/trace
EXPORTS: (format-socks4-response granted portno addr)
         (parse-socks4-request str)
         (spawn-socks4-proxy portno auth-fn conn-fn)
```

3.1.33 lib/checksum-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.34 lib/args-fu

```
IMPORTS:
EXPORTS: (args-fu head keys tail terms)
         (head-fu count terms)
         (key-fu keys terms)
         (parse-fu head keys tail terms)
         (tail-fu count terms)
```

3.1.35 lib/optimize

```
IMPORTS:
EXPORTS: (optimize assembly)
```

3.1.36 lib/options

```
IMPORTS:
EXPORTS: (option options key default)
```

3.1.37 lib/waspc

```
IMPORTS: lib/compile, lib/optimize
EXPORTS: (waspc module)
```

3.1.38 lib/patch

```
IMPORTS:
EXPORTS: (patch in out)
         (patch2 left right)
         (preface-connection data conn)
         (preface-input data in)
```

3.1.39 lib/tag-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.40 lib/bridge

```
IMPORTS: lib/object
EXPORTS: <bridge>
         (bridge-lanes bridge)
         (bridge? value)
         (find-reserved-lane bridge tag)
         (input (<lane> lane))
         <lane>
         (lane-recv lane)
         (lane-tag lane)
         (lane-xmit lane)
         (lane? value)
         (make-bridge xmit recv)
         (make-lane bridge)
         (make-reserved-lane bridge)
         (output (<lane> lane))
```

3.1.41 lib/iterate

```
IMPORTS:
EXPORTS: (all fn arg0 argN ...)
         (any fn arg0 argN ...)
         (filter fn seq)
         (find fn seq)
         fold
         (fold-left fn result arg0 argN ...)
         (fold-right fn result arg0 argN ...)
         (for-each fn arg0 argN ...)
         (ifilter fn seq)
         (imap fn arg0 argN ...)
         (input-iter input)
         (integer-range min max)
         (iter value)
         (iter->list iter)
         (join-iters iter0 iterN ...)
         (list-index fn lst)
         (list-iter list)
         (map fn arg0 argN ...)
         (repeat value count)
```

3.1.42 lib/repl

```
IMPORTS: lib/compile, lib/eval, lib/optimize, lib/trace
EXPORTS: (repl)
```

3.1.43 lib/cons-filter

```
IMPORTS: lib/filter
EXPORTS:
```

3.1.44 lib/http-file-server

```
IMPORTS: lib/http-server
EXPORTS: (offer-http-file server path type data)
         (spawn-http-file-server portno)
         (withdraw-http-file server path)
```

3.1.45 lib/import

```
IMPORTS: lib/waspc
EXPORTS: (import path)
```

3.1.46 waspdoc/c-file

```
IMPORTS: waspdoc/source
EXPORTS: (waspdoc-scan-c path)
```

3.1.47 waspdoc/dump

```
IMPORTS:
EXPORTS: (dump-waspdoc-source src)
         (format-defn (<waspdoc-func-defn> defn))
```

3.1.48 waspdoc/dump-source

```
IMPORTS: waspdoc/dump, waspdoc/ms-file
EXPORTS: (dump-module name)
         (dump-source path)
```

3.1.49 waspdoc/base

```
IMPORTS:
EXPORTS: (set-waspdoc-root! path)
         (waspdoc-root)
```

3.1.50 waspdoc/source

```
IMPORTS: lib/object
EXPORTS: (add-waspdoc-source-export! source defn)
         (list-waspdoc-source-exports source)
         (make-waspdoc-defn source id)
         (make-waspdoc-func-defn source id formals)
         (make-waspdoc-source id name imports exports)
         (make-waspdoc-type-defn source id)
         (name->id name)
         (path->id path)
         <waspdoc-defn>
         (waspdoc-defn-id waspdoc-func-defn)
         (waspdoc-defn-source waspdoc-func-defn)
         (waspdoc-defn? value)
         <waspdoc-func-defn>
         (waspdoc-func-defn-formals waspdoc-func-defn)
         (waspdoc-func-defn? value)
         <waspdoc-source>
         (waspdoc-source-exports waspdoc-source)
         (waspdoc-source-id waspdoc-source)
         (waspdoc-source-imports waspdoc-source)
         (waspdoc-source-name waspdoc-source)
         (waspdoc-source? value)
         <waspdoc-type-defn>
         (waspdoc-type-defn? value)
```

3.1.51 waspdoc/ms-file

```
IMPORTS: lib/compile, waspdoc/source
EXPORTS: (waspdoc-ms-path? path)
         (waspdoc-scan-ms path)
```

3.1.52 waspdoc/check-source

```
IMPORTS: waspdoc/dump, waspdoc/ms-file, waspdoc/source
EXPORTS: (check-source)
         (check-source-export defn)
         (check-source-exports defns)
         (check-source-file path)
         (create-defn-doc-file doc-id defn)
         (create-func-doc-file doc-id defn)
         (create-source-doc-file doc-id source)
         (create-type-doc-file doc-id defn)
         (defn-doc-id defn)
         (ensure-defn-doc-file defn)
         (ensure-source-doc-file source)
         (find-source-files)
         (source-doc-id source)
         wasp-src-dirs
```

3.1.53 mosref/shell

```
IMPORTS: lib/catch, mosref/console, mosref/format, mosref/node, mosref/parse
EXPORTS: (alert rest ...)
         (bind-mosref-cmd verb usage info impl)
         (do-mosref-cmd shell terms)
         (find-mosref-cmd verb)
         <mosref-cmd>
         (mosref-cmd-impl mosref-cmd)
         (mosref-cmd-info mosref-cmd)
         (mosref-cmd-usage mosref-cmd)
         mosref-cmdq
         (mosref-cmds)
         <mosref-shell>
         (mosref-shell-console mosref-shell)
         (mosref-shell-running mosref-shell)
         mosref-shellq
         (run-mosref-shell console node)
         (set-mosref-shell-console! mosref-shell console)
         (set-mosref-shell-running! mosref-shell running)
```

3.1.54 mosref/node

```
IMPORTS: lib/args-fu, lib/bridge, lib/filter, lib/object, lib/package-filter, lib/with-io,  ↩
    mosref/format
EXPORTS: (clear-node-prop! node key)
         <console-node>
         (console-node? value)
         <drone-node>
         (drone-node-bridge drone-node)
         (drone-node-ecdh drone-node)
         (drone-node-link drone-node)
         (drone-node-sin drone-node)
         (drone-node? value)
         (expect-data recv)
         (expect-signal recv)
         (expect-succ recv)
         (find-drone-by-bridge bridge)
         (find-mosref-node id)
         (find-node-prop node key)
```

```
         (format-propval key src)
         (has-node-prop? node key)
         (list-mosref-nodes)
         (list-node-props node)
         (make-console-node addr portno)
         (make-drone-node id link sin ecdh)
         <node>
         (node-id drone-node)
         (node? value)
         (register-prop key alts help valid fmt)
         (resolve-key alt)
         (set-node-bridge! node bridge)
         (set-node-prop! node key val)
         (spawn-node-program (<drone-node> node) program)
         (validate-prop key src)
```

3.1.55 mosref/transport

```
IMPORTS: lib/bridge, lib/buffer-input, lib/patch, lib/trace, lib/with-io
EXPORTS: (decrypt keystr plaintext)
         (encrypt keystr plaintext)
         find-public-key
         find-shared-secret
         (make-iv)
         (make-mosref-recv extra recv recv-key recv-iv)
         (make-mosref-xmit xmit xmit-key xmit-iv)
         (make-private-key)
         (spawn-endpoint endpoint break xmit recv)
```

3.1.56 mosref/props

```
IMPORTS: mosref/prop/address, mosref/prop/online, mosref/prop/platform, mosref/prop/port
EXPORTS:
```

3.1.57 mosref/drone

```
IMPORTS: lib/buffer-input, lib/crypto-filter, lib/iterate, lib/package-filter, lib/tag- ↩
    filter, lib/with-io, mosref/transport
EXPORTS: (drone-affiliation console-public xmit recv)
         drone-bridge
         (drone-broken bridge)
         (drone-endpoint bridge)
```

3.1.58 mosref/cmd/set

```
IMPORTS: mosref/node, mosref/props, mosref/shell
EXPORTS: (format-property key-val)
         (parse-property key-val)
```

3.1.59 mosref/cmd/clear

```
IMPORTS: mosref/node, mosref/shell
EXPORTS:
```

3.1.60 mosref/cmd/nodes

```
IMPORTS: mosref/node, mosref/shell
EXPORTS:
```

3.1.61 mosref/cmd/drone

```
IMPORTS: mosref/listener, mosref/node, mosref/shell
EXPORTS: (node-make-sin (<console-node> node) portno)
         (node-sin-listen (<console-node> node) portno sin)
         (spawn-drone-listener node next)
```

3.1.62 mosref/cmd/exploit

```
IMPORTS: mosref/shell
EXPORTS:
```

3.1.63 mosref/cmd/help

```
IMPORTS: mosref/shell
EXPORTS:
```

3.1.64 mosref/cmd/cp

```
IMPORTS: mosref/node, mosref/shell
EXPORTS: (get-node-file (<drone-node> node) path)
         (put-node-file (<drone-node> node) path data)
```

3.1.65 mosref/cmd/recover

```
IMPORTS: mosref/cmd/drone, mosref/listener, mosref/node, mosref/shell
EXPORTS:
```

3.1.66 mosref/cmd/sh

```
IMPORTS: mosref/node, mosref/shell
EXPORTS: node-shell-prog
         (spawn-node-shell (<console-node> node) cmd)
```

3.1.67 mosref/cmd/load

```
IMPORTS: mosref/cmd/do, mosref/shell
EXPORTS:
```

3.1.68 mosref/cmd/proxy

```
IMPORTS: lib/socks-server, lib/tcp-server, mosref/node, mosref/shell
EXPORTS: (node-tcp-connect (<drone-node> node) addr portno lapse)
         (spawn-node-proxy node portno secret)
```

3.1.69 mosref/cmd/do

```
IMPORTS: mosref/node, mosref/shell
EXPORTS: (eval-node-expr (<drone-node> node) expr)
```

3.1.70 mosref/cmd/with

```
IMPORTS: mosref/node, mosref/props, mosref/shell
EXPORTS:
```

3.1.71 mosref/cmd/on

```
IMPORTS: mosref/shell
EXPORTS:
```

3.1.72 mosref/cmd/exit

```
IMPORTS: mosref/shell
EXPORTS:
```

3.1.73 mosref/parse

```
IMPORTS: mosref/format
EXPORTS: (opt-integer terms for)
         (opt-term terms)
         (parse-flag val)
         (parse-host-spec s)
         (parse-port s)
         (parse-port-spec s)
         (req-node terms for)
         (req-path current-node terms for)
         (req-term terms what ...)
```

3.1.74 mosref/listener

```
IMPORTS: lib/buffer-input, lib/patch, lib/record
EXPORTS: (make-mosref-sin portno)
         (mosref-sin-listen portno sin)
```

3.1.75 mosref/prop/platform

```
IMPORTS: lib/build, mosref/format, mosref/node, mosref/parse
EXPORTS: (node-platform node)
         (set-node-platform! node value)
```

3.1.76 mosref/prop/address

```
IMPORTS: mosref/format, mosref/node, mosref/parse
EXPORTS: (node-addr node)
         (set-node-addr! node value)
```

3.1.77 mosref/prop/online

```
IMPORTS: mosref/format, mosref/node, mosref/parse
EXPORTS: (node-online node)
         (set-node-online! node value)
```

3.1.78 mosref/prop/port

```
IMPORTS: mosref/format, mosref/node, mosref/parse
EXPORTS: (node-has-port? node)
         (node-port node)
         (set-node-port! node value)
```

3.1.79 mosref/format

```
IMPORTS:
EXPORTS: (format-addr addr)
         (format-flag flag)
         (format-ipv4 i)
         (format-path path)
         (send-err items ...)
         (send-line items ...)
```

3.1.80 mosref/console

```
IMPORTS: lib/bridge, lib/buffer-input, lib/build, lib/crypto-filter, lib/iterate, lib/line- ↩
    filter, lib/package-filter, lib/s-filter, lib/with-io, mosref/transport
EXPORTS: (console-affiliation session-private xmit recv)
         (console-broken bridge)
         (console-endpoint bridge)
         (console-repl xmit recv)
         (make-drone-exe console-addr sin console-portno console-public platform)
```

3.1.81 mosref/cmds

```
IMPORTS: mosref/cmd/clear, mosref/cmd/cp, mosref/cmd/do, mosref/cmd/drone, mosref/cmd/exit, ↩
     mosref/cmd/help, mosref/cmd/load, mosref/cmd/nodes, mosref/cmd/on, mosref/cmd/proxy,  ↩
    mosref/cmd/recover, mosref/cmd/set, mosref/cmd/sh, mosref/cmd/with
EXPORTS:
```

3.1.82 bin/wasp

```
IMPORTS: lib/import, lib/repl, lib/with-io
EXPORTS: (main args ...)
MODULE:  ./bin/mosref
IMPORTS: lib/env, lib/iterate, lib/with-io, mosref/cmds, mosref/console, mosref/shell
EXPORTS: (main args ...)
         (spawn fn args ...)
```

3.1.83 bin/install

```
IMPORTS: lib/build, lib/waspc
EXPORTS: (main args ...)
```

3.1.84 bin/wasp-vim-syntax

```
IMPORTS:
EXPORTS: (write-vim-syntax path)
```

3.1.85 bin/waspc

```
IMPORTS: lib/build
EXPORTS: (main args ...)
         usage
```

3.1.86 bin/waspdoc

```
IMPORTS: waspdoc/check-source, waspdoc/dump-source
EXPORTS: (main args ...)
```

3.1.87 core/module

```
IMPORTS:
EXPORTS: (find-module-file filename)
         (import key)
         (imported? key)
         (load path)
         (load-mo path)
         (module key)
         (preload keys ...)
         (read-module-source path)
         waspvm
```

3.1.88 core/macro

```
IMPORTS:
EXPORTS: (set-macro! key fn)
         (waspvm-syntax)
```

3.1.89 core/file

```
IMPORTS:
EXPORTS: (convert-path path)
         (path-join items ...)
         (read-data-file path)
         (read-lisp-file path)
         (write-data-file path content)
         (write-lisp-file path data)
```

3.1.90 core/config

```
IMPORTS:
EXPORTS: (set-site-config! key value)
         (site-config key)
         waspvm
```

3.1.91 core/io

```
IMPORTS:
EXPORTS: (input [any]) => <input>
         (output [any]) => <output>
         (print* items ...)
         (println* items ...)
         (send message [dest])
         (tcp-connect args ...)
         (timeout (<integer> ms) [any])
         timeout-input
         (wait) => any
         (wait src) => any
         (wait (<integer> ms)) => any
         (wait (<integer> ms) src) => any
```

3.2 Primitives

3.2.1 channel

```
(send-output data [output])
(wait-input [input])
```

3.2.2 connection

```
(make-connection input)
(connection-input connection)
(connection-output connection)
```

3.2.3 core

```
(string->uppercase string)
(string->lowercase string)
(string-read-expr! string)
(exprs->string list)
(xml-escape string)
(percent-encode (<string> data) (<string> mask))
(percent-decode (<string> data))
(string->byte (<string> data))
(string->word (<string> data))
(string->quad (<string> data))
(string->integer (<string> s))
(vector->list (<vector> v))
(list->vector (<list> l))
(cadr (<pair> p))
(reverse (<list> l))
(reverse! (<list> l))
(caddr (<pair> p))
(equal? arg0 ...)
(not x)
(last-pair (<list> l))
(last-item (<list> l))
(list-ref (<list> l) (<integer> n))
(list-refp (<list> l) (<integer> n))
(abs (<integer> n))
(* (<ingeger> n) ...)
(/ (<ingeger> n) ...)
(quotient (<integer> n1) (<integer> n2))
(remainder (<integer> n1) (<integer> n2))
(number->string (<integer> n))
(string->symbol (<string> s))
(symbol->string (<symbol> s))
(make-vector (<integer> n) [init])
(list-index item (<list> l))
(memq item (<list> l))
(member item (<list> l))
(exit [(<integer> code)])
(equal? arg0 ...)
(eq? arg0 ...)
(list? obj)
(integer? obj)
(cons obj1 obj2)
(car (<pair> p))
(cdr (<pair> p))
(set-car! (<pair> p) obj)
(set-cdr! (<pair> p) obj)
(vector ...)
(vector-ref (<vector> v) (<integer> n))
(vector-set! (<vector> v) (<integer> n) any)
(vector-length (<vector> v))
(string-length (<string> s))
(substring (<string> s) (<integer> index) (<integer> length))
(string-head (<string> s) (<integer> length))
(string-tail (<string> s) (<integer> index))
(string-ref (<string> s) (<integer> index))
(string-set! (<string> s) (<integer> index) (<integer> byte))
(string-fill! (<string> s) (<integer> index) (<integer> length) (<integer> byte))
(= (<integer> arg0) ...)
(< (<integer> arg0) ...)
(> (<integer> arg0) ...)
(<= (<integer> arg0) ...)
(>= (<integer> arg0) ...)
```

```
(!= (<integer> arg0) ...)
(string=? (<string> arg0) ...)
(length (<list> l))
(error-key (<error> e))
(error-info (<error> e))
(error-context (<error> e))
(map-car (<list l))
  equivalent to (map car l)
(map-cdr (<list> l))
  equivalent to (map cdr l)
(thaw (<string> s))
(freeze obj)
(string-append (<string-or-int> s) ...)
(assq obj (<list> l))
(assoc obj (<list> l)
(getcwd)
(chdir (<string> s))
(argv [(<integer> index)])
(argc)
(refuse-method)
(get-global (<symbol> s) default)
(enable-trace)
(disable-trace)
(make-tc ...)
(tc-clear! (<tc> t))
(tc-append! (<tc t) (<list l))
(tc-next! (<tc> t))
(tc-empty? (<tc> t))
(tc-add! (<tc> t) item ...)
(tc-remove! (<tc> t) item ...)
(tc-prepend! (<tc> t) item)
(tc->list (<tc> t))
(string->exprs (<string> s))
(globals)
(make-set obj ...)
(set-add! (<set> s) obj ...)
(set-remove! (<set> s) obj ...)
(set-member? (<set> s) obj ...)
(set->list (<set> s))
(make-dict (<list-of-pairs> l))
(dict->list (<dict> d))
(dict-keys (<dict> d))
(dict-values (<dict> d))
(dict-set? (<dict> d) key)
(dict-set! (<dict> d) key value)
(dict-ref (<dict> d) key alternate)
(dict-remove! (<dict> d) key)
(string-find (<string> s) (<string> item))
(string-begins-with? (<string> s) (<string> item))
(strip (<string> s))
(strip-head (<string> s))
(strip-tail (<string> s))
(string-ends-with? (<string> s) (<string> item))
(split-lines (<string> s))
(string-split (<string> s) (<string> item))
(string-replace (<string> s) (<string> pattern) (<string> replacement))
(string-split* (<string> s) (<string> item))
(string-join (<string> sep) (<string> item) ...)
(function? obj)
(function-name (<function> f))
(make-string [(<integer> capacity)] [(<integer> init)])
(flush-string (<string> s))
```

```
(empty-string? (<string> s))
(string-skip-space! (<string> s))
(string-skip! (<string> s) (<integer> offset))
(string-read! (<string> s) [(<integer> max)])
(string-append-byte! (<string> s) (<integer> byte))
(string-read-byte! (<string> s))
(string-read-line! (<string> s))
(string-append-word! (<string> s) (<integer> word))
(string-read-word! (<string> s))
(string-append-quad! (<string> s) (<integer> quad))
(string-read-quad! (<string> s))
(string-alter! (<string> s) (<integer> offset) (<integer> length) (<string> data))
(byte->string (<integer> byte))
(word->string (<integer> word))
(quad->string (<integer> quad))
(string-prepend! (<string> s) (<string-or-integer> data))
(append (<list> l) ...)
(append! (<list> l) ...)
(string-append! (<string> s) (<string-or-integer> data))
(string-erase! (<string> s) (<integer> offset) (<integer> length))
(string-insert! (<string> s) (<integer> offset) (<string-or-integer> data))
(copy-string (<string> s))
```

3.2.4 crc32

```
(crc32 (<string> s))
```

3.2.5 curve

```
(curve25519-public (<string> private)) => (<string> public)
(curve25519-secret (<string> private) (<string> public)) => (<string> s)
```

3.2.6 error

```
(error (<string> key) ...)
(traceback (<error> e) [(<string-or-output> dest)])
(re-error (<error> e))
```

3.2.7 file

```
(open-file (<string> path) (<string> flags) [(<integer> mode)]) => (<file> result)
(file-len (<file> f)) => (<integer> r)
(read-file (<file> f) (<integer> n)) => (<string> s)
(close-file (<file> f))
(closed-file? (<file> f)) => (<boolean> r)
(write-file (<file> f) (<string> data))
(file-skip (<file> f) (<integer> n)) => (<integer> r)
(file-pos (<file> f)) => (<integer> r)
(file-seek (<file> f) (<integer> n)) => (<integer> r)
*path-sep* => (<string> s)
*line-sep* => (<string> s)
```

3.2.8 filesystem

```
(path-mtime (<string> path)) => (<integer> r)
(locate-path (<string> filename) (<string> path0) ...) => (<string-or-bool> r)
(path-exists? (<string> path)) => (<boolean> r)
(dir-path? (<string> path)) => (<boolean> r)
(file-path? (<string> path)) => (<boolean> r)
(dir-files (<string> path)) => (<list> r)
(rename-file (<string> old-path) (<string> new-path))
(remove-file (<string> path))
```

3.2.9 list

```
(list item0 ...) => (<list> items)
```

3.2.10 memory

```
(null? obj) => (<boolean> r)
```

3.2.11 multimethod

```
(make-multimethod (<list-or-true> signature) (<function> pass) (<function> fail)) => (< ↩
    multimethod>)
(isa? value type) => (<boolean>)
```

3.2.12 number

```
(+ (<integer> n0) ...) => (<integer>)
(- (<integer> n0) ...) => (<integer>)
(& (<integer> n0) ...) => (<integer>)
(| (<integer> n0) ...) => (<integer>)
(^ (<integer> n0) ...) => (<integer>)
(<< (<integer> base) (<integer> offset)) => (<integer>)
(>> (<integer> base) (<integer> offset)) => (<integer>)
(! (<integer> base)) => (<integer>)
*max-int* => (<integer>)
*max-imm* => (<integer>)
*min-int* => (<integer>)
*min-imm* => (<integer>)
```

3.2.13 os

```
(resolve-ipv4 (<string> address)) => (<integer>)
(conio-size) => (<list>)
(conio-goto (<integer> row) (<integer> col))
(conio-clear)
(conio-cls)
(tty-size tty) => (<list>)
(start-tcp-connect host service) => (<connection>)
(os-connection-input (<connection c)) => (<input>)
(os-connection-output (<connection c)) => (<output>)
(serve-tcp (<integer> port)) => (<os-service>)
```

```
(close-service (<os-service> s))
(scan-io)
(reset-console-colors)
(set-console-colors (<integer-or-false> fg) (<integer-or-false> bg))
(console-blit (<string> text) (<string> fg) (<string> bg) [(<integer> offset) (<integer>  ↩
    length)])
(unbuffer-console)
*console* => (<connection>)
```

3.2.14 plugin

```
(load-subsystem (<string> path) (<string> init))
*plugin-ext* => (<string>)
```

3.2.15 print

```
(print (<string> value))
(format (<string> value) [(<integer> breadth) (<integer> depth) (<string> buffer))
```

3.2.16 procedure

```
(assemble (<list> source)) => (<procedure>)
```

3.2.17 process

```
(spawn (<function> f) ...) => (<process>)
(halt)
(current-input) => (<input>)
(current-output) => (<output>)
(process-input (<process> p)) => (<input>)
(process-output (<process> p)) => (<output>)
(set-current-input! (<input> i))
(set-current-output! (<output> o))
(set-process-input! (<process> p) (<input> i))
(set-process-output! (<process> p) (<output> o))
(current-process) => (<process>)
(dump-actives)
```

3.2.18 queue

```
(make-queue) => (<queue>)
(queue-input (<queue> q)) => (<queue-input>)
(queue-output (<queue> q)) => (<queue-output>)
```

3.2.19 regex

```
(match-regex (<regex> r) (<string> text) [(<string> flags)]) => (<tc-or-false>)
(match-regex* (<regex> r) (<string> text) [(<string> flags)]) => (<pair-or-false>)
(make-regex (<string> pattern) [(<string>) flags])
(string-read-regex! (<string> text) (<regex> r) [(<string> flags)]) => (<any>)
```

3.2.20 salsa

```
(make-salsa20-key (<string> seed) [(<string> iv)]) => (<salsa20-key>)
(salsa20-encrypt (<salsa20-key> key) (<string> plaintext) [(<string> iv)]) => (<string>)
(salsa20-decrypt (<salsa20-key> key) (<string> ciphertext) [(<string> iv)]) => (<string>)
(read-entropy (<integer> amount)) => (<string>)
(random-integer (<integer> min) (<integer> max)) => (<integer>)
(read-prng (<integer> amount)) => (<string>)
```

3.2.21 shell

```
(spawn-command (<string> path) [(<list> args) (<list> env)]) => (<connection>)
(run-command (<string> command)) => (<integer>)
*environ* => (<list>)
```

3.2.22 tag

```
(type-name value) => (<symbol>)
(type value) => (<cell-or-value>)
(repr value) => any
(tag (<cell> c)) => (<tag>)
(make-tag (<string> name) ...) => (<tag>)
(tag-info (<tag> t)) => any
(cell (<tag> t) value) => (<cell>)
```

3.2.23 time

```
(timeout (<integer> ms) (<input> input)) => (<task>)
(cancel-timeout (<task> t))
(pause [(<integer> ms)])
```