***Utilities*** ***Reference .***

**version *0.2.11***

***mu-sys .***

*Name*: mu-sys – *minimal exec command*

*Synopsis*: mu-sys [*options...*] [*file...*]

*Options*: 0.0.2

-v print version string and exit

-l *SRCFILE* load SRCFILE in sequence

-e *SEXPR* evaluate SEXPR and print result

-q *SEXPR* evaluate SEXPR quietly

-c *JSON* environment configuration

[file ...] load source file(s)

***mu-server .***

*Name*: mu-sys – *minimal exec command*

*Synopsis*: mu-sys [*options...*] [*file...*]

*Options*: 0.0.2

-v print version string and exit

-l *SRCFILE* load SRCFILE in sequence

-e *SEXPR* evaluate SEXPR and print result

-q *SEXPR* evaluate SEXPR quietly

-c *JSON* environment configuration

[file ...] load source file(s)

***mu-exec .***

*Name*: mu-sys – *minimal exec command*

*Synopsis*: mu-sys [*options...*] [*file...*]

*Options*: 0.0.2

-v print version string and exit

-l *SRCFILE* load SRCFILE in sequence

-e *SEXPR* evaluate SEXPR and print result

-q *SEXPR* evaluate SEXPR quietly

-c *JSON* environment configuration

[file ...] load source file(s)

***manifest .***

*Name*: manifest – system development tool

*Synopsis*: manifest *command* [*options…*]

*Commands*: 0.0.16

*help* this message

*version* *manifest* version

*init* init ./ as workspace

*env* print dev environment

*clean* clean all artifacts

*build release* build system

*profile*

*debug*

*image* *build -–*out=*path* |

[--image=*path* | --config=*config*]

\*[--load=*path* | --eval=*sexpr*]]

*view* --image=*path*

manage heap images

*symbols* *reference* [--module=*name*] |

*crossref* [--module=*name*] |

*metrics* [--module=*name*]

symbol reports

*install* install release system-wide, may

need sudo(8)

*commit* *rustfmt* and *clippy*,

pre-commit linting

*test* run regression test suite

*bench* *base* [--ntests=*number*]

*current* [--ntests=*number*]

*footprint* [--ntests=*number*]

run benchmarks

*General options*:

--verbose verbose operation

***mu-listener .***

*Name*: mu-listener – *mu* system REPL

*Synopsis*: mu-listener

*Description*: 0.0.2

*mu-listener* is a generalized command-line REPL that can

be configured to load and run in any of the *mu* namespaces.

It is intended for debugging and exploration during

development.

*mu-listener* has no command-line arguments, it is configured

by a *JSON*-format dotfile named *.mu-listener.* An example

dotfile can be found in */opt/mu/lib/mu-listener*. If a

*.mu-listener* is not found in the current directory, the

user’s home directory will be searched and if found there

will be used. *mu-listener* does not require a *.mu-listener*

file.

{

"config": "npages: 2048",

"namespace": "core",

"rc": "mu-listener.rc"

}

The configuration syntax can specify three options, none

of which are required.

The *config* option supplies a *mu* Env configuration

specification in *JSON* format, see the *mu-ref* refcard

*environment* section for details. If this option is not

specified, the *mu* Env will be created with default values.

The *namespace* option supplies the namespaces that should be

loaded, see the *mu-ref* refcard *namespaces* section for

details. If this option is not specified *mu-listener* will

load only the *mu* namespace.

The *rc* option supplies the name of a source file to be

loaded before the REPL runs. Additional symbols may be

supplied there.