Medley Interlisp

2023 Project Update

18 March 2023

Interlisp.org

What is Medley Interlisp?

- The software for the Xerox Lisp machines
developed from 1960’s (BBN), thru 1970’s and 80’s (Xerox), and 90’s (Envos, Venue)
- ACM Software System Award 1992

The features of ***structure editing****,* ***source code management****,* ***code a***
nalysis, cross- ***referencing*** *combined to support* ***rapid incremental development****.*
The 1992 ACM Software System Award was awarded to the Interlisp system for pioneering wo
rk in programming environments.

- Then: expensive, slow, unwieldy... uvavailable

- Now: open source, 1000 times faster, on modern hardware of all si
zes... *enjoy*

Who might use Medley Interlisp?

- Retrocomputing enthusiasts

- Researchers, IDE tool creators, looking into ideas

- Software and AI historians (present and future)

- Software archivists (SPN, technical infrastructure)

- Developers of new applications

What have we been doing?

+ Adapt the system to current environments and standards

    hardware: keyboard, mouse, display; 64bit, little endian

    software: standard C, Unicode, Posix, UI Guidelines

+ Lower barriers to entry

+ Demonstrate unique and original features

+ Help revive other applications built using system

+ Improve maintainability for future users

+ Gather and update documentation

Who is involved?

+ Some original developers from Xerox PARC (from 70's and later)

+ Developers of s using Interlisp (recursive revivers)

- Open Source contributors

- Software History and Future Technology/ California State University Channel Islands

+ Friends, enthusiasts, and ... you?

```
(OPEN-URL "https://github.com/orgs/interlisp/people")
(MEDLEY-CONTRIB "medley")
(MEDLEY-CONTRIB "maiko")
(MEDLEY-CONRIB "online")
(MEDLEY-CONTRIB "Interlisp.github.io")
```

Medley Interlisp is an IDE

- Residential programming for incremental development
  write, debug, edit in the live environment.

In #InterLisp, you could refer to a function you hadn't written
yet, run your codeuntil it
broke, and from the break have it pop open an editor with the sig
nature of your function. You
could also inspect the values of the arguments passed. You could
then write the function and
continue the computation.

Similarly if your code broke with an error, you could edit the fun
ction on the stack to correct
the error, and continue the computation. ...

I remember demonstrating this capability in Interlisp to a very s
mart Python programmer I
know and his response was along the lines of "why would I ever
want that?" Same with the
functionality of the CLOS browser. If you've never used an envi
ronment focused on
programmer productivity you have no idea how to even think al
ong those lines.

It's like we've gone back to the stone age.

Simon Brooke

Incremental Development

- Define, edit, debug in the live environment

  ... write out to files to save for future session
- System tracks what changed

spelling correction

```
(DEFINEQ (FOO (X) (PLUSS X X]
PP FOO
(FOO 6]
y
PP FOO
```

file manager

```
FILES?)
y BAR
(CLEANUP)
SEE BAR
```

History

```
??
UNDO <dwim event>
PP* FOO
FIX DEFINEQ   <change to (+ X PARAM) >
(FOO 6)
RETRY
```

- Helpsys and DInfo interactive **docs**

```
MAN INFILE
MAN CL:WITH-OPEN-FILE
MAN CLHS.OPENER
<right click DInfo>
```

Masterscipe (interactive cross reference)

```
LOAD(HELPSYS PROP)
. ANALYZE ON HELPSYS
. SHOW WHERE ANY CALLS CLHS.OPENER
USE EDIT FOR SHOW
. SHOW PATHS FROM HELPSYS
<load in fuller.database>
```

Common Lisp and Interlisp Integration

an Inter- medley of Uncommon Lisps

- freely intermix CL and IL function, macro, variable definitions/declarations

- All Datatypes are common: lists, NIL, symbols, arrays, strings, structures, numbers

- many functions and special forms are the same: CAR, CDR, CONS, COND

- Some things are slightly different: CL:EVAL vs. IL:EVAL, CL:LAMBDA vs. IL:LAMBDA, ...

- Extend CL to include IL and IL to include CL

*Demo: Look at History ED(FOO), change *PACKAGE*, look at changes; edit it to use CL:*

Structure editors

`EDITMODE(TELETYPE)` original structure editor: powerful, programmable, short commands (BO 3) P (SW 1 2). Useful for editing huge list structures or Lisp functions.

`EDITMODE(DEDIT)` (Load it to try). Like the TTY editor with menu and showing the results in a window.

`EDITMODE(SEDIT:SEDIT)` Default editor. SEdit maintains illusion that yo
u are editing structure. Parentheses are balanced at all times, but you can just type and backspace. Keyboard controls and attached menu.

Common to all: User never counts parentheses, modifies whitespace, or needs to fiddle with line breaks or indentation

Medley Interlisp is an OS

- Interlisp-D was the **whole operating system**:

  scheduler, window manager, network, drivers

- Now can rely on host OS for device drivers

ARCHITECTURE MAIKO / MEDLEY

- D-machines had microcode to interpret a byte-coded insruction set

- The images are portable -only the emulator is machine/os specific

- **Medley is small** (relatively speaking). Bytecode are compact. CDR-coding is effective. Medley online uses 16MB/user (64MB max). Installed, 256mb max. Time to make new image 15 seconds; restart a saved image in a blink.

Demo: show stats?

Revive LispUsers, Library, Internal

With Sources there are four places:

SoOther development tools: SPY, File Browser

GitHub GITFNS

Revive Applications using Interlisp

Notecards - early HyperText

Rooms - Screen / Desktop management

LOOPS - Lisp Object Oriented Programming System (not CLOS)

Truckin' - LOOPS game for teaching "Knowledge Representation"

LFG - Lexical Functional Grammar

.... and others

Modernizing Medley

Goals

- Adapt Medley UI to current hardware

- Make Medley work more like current applications for comfort

Mouse: Wheel Scroll, Window move on title, Window resize on corner. Warning: Exec windows. Consolation: you can do it all with menus. Reduction: some menus are invoked by "middle" mouse button which on some mice is hard to press.

Keyboard: Desired state: compatible keystrokes. Subtasks: decide what keystrokes do what; slash through umpteen layers of keyboard re-iterpretation.

Display: Color, high-resolution displays, modern fonts.

Reduce Barriers to Entry

- Running Online

```
OPEN-URL("https://www.youtube.com/watch?v=mI3Ga5LyIlI")

OPEN-URL("https://online.interlisp.org")
```

- Installation Instructions

```
OPEN-URL("https://interlisp.org/running/")
```

Running on Modern Systems

- about K&R C and C standards; type declarations, big-endian vs. little endian, 32 bit vs 64 eliminating errors, file-sytem changes, process handles, bugs left over from 24- to 28-bit address space

- docker, installers, CO/CI(?), two levels of virtualization

- online.interlisp.org: connect in seconds. Sessions

- Mysteries in the code we inherited, "software archeology"

- Robust "loadup process", all day vs seconds

- bytecode virtual machine aids portability.

```
fb {li}<maiko>bin/makefile-*.*-x
```

Linux, Mac, Windows (WSL1, WSL2, Cygwin)
Docker, FreeBSD, OpenBSD, SunOS5
x86_64, i386, arm7l, arm64, and older

Getting Involved

- Try things out, report new problems, or new reproducible cases

```
(OPEN-URL "https://github.com/interlisp/medley/issues")
```

```
(FILESLOAD PICK)
```

```
PICK ISSUE
```

- Testers, common lisp implementers?

- Maker projects? SoC + eInk projects?

- Network emulation