Who is this fellow, anyway?

• Worked on OpenMCL / Clozure CL since 2007 (both as a Clozure Associates employee and now independently)

• First used Allegro CL on the NeXT machine, and then used MCL
This Old Lisp
http://turnoff.us/geek/developers/
as modified at https://twitter.com/nihirash/status/880829816072802304
Clozure CL background
<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1958</td>
<td>Lisp</td>
</tr>
<tr>
<td>1984</td>
<td>Common Lisp (CLTL1)</td>
</tr>
<tr>
<td>1987</td>
<td>Coral Common Lisp 🌍 1MB Macintosh Plus</td>
</tr>
<tr>
<td>1988</td>
<td>Macintosh Common Lisp Apple acquires Coral</td>
</tr>
<tr>
<td>Year</td>
<td>Event 1</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1994</td>
<td>MCL transferred to Digitool</td>
</tr>
<tr>
<td>1995</td>
<td>MCL ported to PowerPC</td>
</tr>
<tr>
<td></td>
<td>MCL (without GUI/IDE) ported to VxWorks and LinuxPPC</td>
</tr>
<tr>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>OpenMCL</td>
</tr>
</tbody>
</table>
port to 32-bit ARM
port to Darwin (macOS)
port to Windows
port to FreeBSD
Objective-C interface
port to 64-bit PowerPC
port to x86-64
port to 32-bit x86
native threads
port to Solaris-ish
and more...
In 2007, Alice Hartley of Digitool announced that the code for the original MCL would be open sourced (under the LLGPL).

Thus, to avoid confusion between OpenMCL and the newly open-sourced MCL, OpenMCL was renamed to Clozure CL

As a bonus, this made the CCL package name make sense again.
Digression:

Why did Digitool throw in the towel on MCL?
Closure

Clojure

Clozure

Closzjure?

🤔
Clozure CL today
• general purpose implementation

• targets x86, x86-64, ARM (ppc32, ppc64 not supported after release 1.10)

• runs on Linux, macOS, FreeBSD, Solaris, Windows
It’s old

;;; from slisp reader2.lisp, and apparently not touched
;;; in 20 years.

(defun parse-integer (string &key (start 0) end
  (radix 10) junk-allowed)
  ...)

fancy loop macro, pretty printer, format, etc.
old subproblems

- printing and reading floating-point numbers
- bignum operations
- disassembler
- random number generation
Who uses it?
Multiple constituencies

• “batch” users on large memory machines
• hackers using Emacs and SLIME
• macOS Cocoa IDE users (Mac App Store or otherwise)
Some CCL technologies

- compiler
- garbage collector
- threads
- FFI
THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILING!

OH. CARRY ON.

https://xkcd.com/303/
• Generates reasonable code quickly.

• With some effort with declarations, floating-point code can be halfway decent.

• It could afford to work a little harder and still be fast.
more compiler

• builds CCL itself in under a minute

• new users uncover new categories of bugs (including performance bugs)
Native threads

• use multiple cpu cores

• hash tables

• streams

• thread-local shallow binding for special vars
Single-space compacting GC

old objects

young objects

generations
GC implications

- Objects may move at any time
- Passing data to foreign code generally requires copying
- current GC stops other threads
Convenient FFI

It’s easy to call C functions if you know their names.

? (external-call "getpid" :pid_t)
4771
more FFI

There is notation to describe and access foreign data.

(rletz ((x (:array :double 10)))
  (setf (paref x (:array :double) 5) 100d0
  (paref x (:array :double) 8) -1000d0)
  (external-call "cblas_idamax" :int 10
   (* double) x :int 1 :int))
Interface translator

Interface translator (based on gcc or libclang) turns .h files into s-expression representation.

CURL_EXTERN CURLcode curl_global_init(long flags);

(function ("/usr/include/curl/curl.h" 2143) "curl_global_init"
(function ((long ())) (typedef "CURLcode") (extern)))
FFI reader macros

Lisp code parses the s-expression data and makes a database used by reader macros. This way, you don’t have to specify foreign types, because they are known from the database.

? (open-shared-library "libcurl.dylib")
#<SHLIB /usr/lib/libcurl.dylib #x30200091FF6D>
? (/_curl_global_init #$CURL_GLOBAL_DEFAULT) 0
related projects

- Test suite based on Paul Dietz’s ANSI CL tests (github.com/Clozure/ccl-tests)
- documentation written using CCLDoc system (github.com/Clozure/ccldoc)
- updated libclang-based ffigen
future plans

• keep up

• continue work on experimental register allocator (which is opt-in via a special optimize quality in 1.12 development branch)

• port to 64-bit ARM

• fix bugs
future plans

• New macOS IDE

• Emacs and SLIME are perhaps a locally-optimal plateau

• your wish here; get in touch
Who owns CCL?

• Clozure Associates has supported CCL development for many years, but the project has never been Clozure’s product or private playground.

• Copyright obtained from Digitool

• Apache 2.0 license instead of LLGPL
Who hacks on CCL?

- Gary Byers, a great hacker and long-term driving force behind CCL, has retired.
- maybe you?
You can help CCL

- On GitHub: https://github.com/Clozure/ccl
- #ccl on Freenode
- openmcl-devel@clozure.com mailing list
- Do cool stuff
Matthew Garrett
@mjg59

Therapist: So you're afraid that you're letting down people you've never met and who you've given something for free?
Me: Yeah basically

12:33 AM - 10 Sep 2017
You can get help for CCL

• Clozure Associates can offer paid support for Clozure CL

• You can hire me to do anything whatever with Clozure CL
I ❤ Common Lisp and Clozure CL
• The standard is stable, and provides a baseline of much useful functionality

• Multiple CL implementations to choose from

• I like Clozure CL. Maybe you like something else. We can still be friends.
• Built-in support for collections
• Automatic storage management
• Dynamic typing
• First-class functions
• Interactive environment
• Extensibility (functions, classes, syntax, reader)
• Uniform syntax (macros)
language & interactivity

• CL has a built-in assumption that the programming environment is going to be interactive

• e.g., trace, break, update-instance-for-redefined-class
The spirit inside the computer

• early micros said “Ready”

• interactive, incremental approach to programming is great for exploring a new problem domain, or working on a problem that you don’t know how to solve
Counterpoint

• CL’s interactive nature lets you jump right in and start messing around with code, when maybe it would be better to think a bit first.

• Furious activity is no substitute for understanding.
I never look back, darling. It distracts from the now.
“Indeed, one of my major complaints about the computer field is that whereas Newton could say, ‘If I have seen a little farther than others it is because I have stood on the shoulders of giants,’ I am forced to say, ‘Today we stand on each other’s feet.’ Perhaps the central problem we face in all of computer science is how we are to get to the situation where we build on top of the work of others rather than redoing so much of it in a trivially different way.”

Richard Hamming
Hmm, I can’t decide on my new year’s resolution...

I could choose a single new JS Framework, and I work with it all year to really learn it properly...

Jquery for example?

Or maybe I should commit to trying a different JS framework every week for the rest of the year, so I stay up-to-date.

The fact that both ideas seem entirely reasonable demonstrates just how f*cked front-end dev is right now.

Hum...

Four well-defined directions
Brushes and chisels
The enjoyment of one’s tools is an essential ingredient of successful work.
“For years, CCL has been the Lisp of choice for performing hardware verification with ACL2. The hash cons / static cons tables make it particularly adept at analyzing the Verilog itself.”
En garde, Lisp naysayers!
Thank you.
Let’s hack more Lisp.

rme@acm.org