

Common Lisp : An Introduction

Ram Vedam

10-23-2017

What we will cover

- What is Common Lisp?
- History
- Getting Started
- Ecosystem Overview
- Basic Language Overview
- Questions

What we will NOT cover

- Macros
- CLOS
- MetaObject Protocol
- Gray Streams
- Packages
- Conditions

What is Common Lisp?

- Direct Descendent of LISP, invented by John McCarthy in 1958
- multi-paradigm language
 - procedural programming
 - functional programming
 - object-oriented programming
 - generic programming
 - can extend Lisp to embrace new paradigms (via macros)
- Functional Programming dominant paradigm (i.e. most code will utilize FP in some way)
- Interpreted as well as compiled language
 - most implementation compile functions by default
- compiler that can create executables and available in REPL

- Commercial
 - Allegro CL (Franz Inc.)
 - LispWorks
 - mocl (compiles to iOS and Android)
- Open Source
 - Armed Bear Common Lisp (ABCL): runs on JVM
 - SBCL (Steel Bank Common Lisp)
 - forked from CMUCL with some added fixes and extensions
 - available on all major platforms
 - CMUCL
 - Clozure CL
 - good Objective-C interoperability
 - ECL
 - Clasp (LLVM)

A Brief History

- Late 1970s
 - Different Lisps available for different computer architectures
 - S-1 Lisp (for Mark II SuperComputer)
 - MACLisp (from Project MAC not Macintosh) and it's derivatives (NIL, BBN Lisp, etc.)
 - InterLisp (started BBN, expanded later Xerox PARC)
 - many others
 - Each Lisp expanded on the original Lisp 1.5 implementation and MACLisp just a little differently
 - Each Lisp provided additional features (Parallel Programming, abstractions to implement OO, etc.)

A Brief History

- 1981: Start of Standardization
 - instigated by DARPA to create a single community Lisp standard
 - took almost 13 years to standardize (ANSI standard official in 1994)
 - CLtL (written by Guy Steele for example)
 - first published in 1984, republished in 1989 (with new additions)

Why is this history necessary?

- CL spec classical example of design by committee
- CL was a "consolidation effort"
 - A lot of compromises were made for reasons of backwards compatibility
 - A lot of decisions were well-discussed and
- Consequences
 - Certain pieces of what is considered "Standard Common Lisp" are not part of ANSI Spec
 - MOP, Gray Streams, Regular Expressions
 - Common Lisp actively developed on and extended through Platform extensions

Getting Started

There are several ways of getting started:

- downloading a Lisp (SBCL, CCL, etc.)
 - Follow the appropriate Lisp Compiler's website on how to best install the distro
 - **Suggestion:** use SBCL unless you have a good reason to try other distros
- downloading and installing Quicklisp
 - located at <https://www.quicklisp.org/beta/>
 - single file download
 - Just load into your REPL, and follow the instructions

Roswell

- Utility to keep track and maintain multiple lisps
- Supported on all Major Operating Systems
 - Linux (via linuxbrew)
 - Mac OS X (via Homebrew)
 - Windows (installer available on Roswell wiki (<https://github.com/roswell/roswell/wiki>) in the Installation Section)
- Best way to get up and running with a lisp
- Best way to maintain and keep track of multiple lisps installed

Quicklisp

- De facto Package Manager for Common Lisp
- Nice curated set of libraries
- Installed by default when using Roswell with different lisp implementations

Every Common Lisp System consists of 4 main layers:

- ANSI Common Lisp
- De facto standards
- Platform Extensions
- Third Party Libraries

ANSI Common Lisp

- the standardized foundation
- EVERY CL compiler implements this standard (at least)

Defacto standards

- libraries that aren't part of the standard but most major implementations implement
- MOP, Gray streams, CFFI, ASDF, etc.

Platform Extensions

- Threading
- Extensible Sequences
- Extensions to leverage processor-specific instructions (VOPs)
- MOP extensions
- Tooling
 - Profilers
 - Advanced Debuggers

Third-Party Libraries

- User Interface Libraries (Qtool, CommonQt, McCLIM, LTK)
- Web Frameworks (Caveman2, ningle, Weblocks, etc.)
- Async and Parallel APIs (lparallel, cl-async, etc.)
- and many more!

Language Overview

Recommendations

- prefer defparameter over defvar
 - defparameter always assigns value to symbol
 - defvar only assigns value once to symbol upon first initialization
 - if using defvar and need to change value of binding using SETF not defvar
- prefer use of lexical environments under normal development
 - few cases where special variables are an exception, but advanced topic

Recommendations

- functional programming by default
 - leads to cleaner, modular, composable code
 - leads into dealing with generic programming (at least in Common Lisp)

Book Recommendation

Introductory Books

- Practical Common Lisp by Peter Siebel
- ANSI Common Lisp by Paul Graham
- Land of Lisp by Conrad Barski

Learning about Macros:

- On Lisp by Paul Graham
- Let over Lambda by Doug Hoyte

Questions??