com.informatimago.common-lisp version 1.2

The packages provided by this library should be PURE conformant Common Lisp packages.

They should compile and run in all Common Lisp compliant implementations, and should have make use of no external package (eg. compatibility library) and no other (implementation dependant) package than COMMON-LISP. They should not use #+/#- to activate or disable implementation specific code. [There remains some packages using #+/#- with implementations specific variants, we're working on removing these forms].

This new version of the *com.informatimago.common-lisp* library has been restructured into various sub-libraries, each with its own ASD file.

This should help managing the dependencies between the individual sub-libraries, and promote the use of the most interesting of them.

com.informatimago.common-lisp.lisp-sexp and *com.informatimago.common-lisp.cesarum* are the two base sublibraries, almost all of the others use them.

Compilation

Starting with version 1.2 of this library, ASDF systems are provided to allow for an easy and modular compilation.

compile.lisp should not be used anymore (until upgraded). *make-depends.lisp* needs an upgrade too.

Compilation with the compile-with-asdf.lisp script

On implementations that don't provide (yet) ASDF, you need to indicate what asdf.lisp file must be loaded, by setting the *cl-user::*asdf-source** variable.

The path to the asdf-binary-locations directory (where the .asd file lies) must be set to the *cl-user::*asdf-binary-locations-directory** variable.

Then the compile-with-asdf.lisp script can be loaded:

```
(in-package :cl-user)

(defvar *asdf-source*
  #P"/data/lisp/packages/net/common-lisp/projects/asdf/asdf/asdf.lisp")

(defvar *asdf-binary-locations-directory*
  #P"/data/lisp/packages/net/common-lisp/projects/asdf-binary-locations/asdf-binary-locations/")

(load "compile-with-asdf.lisp")
```

It will add to *asdf:*central-registry** the current directory and its subdirectories containing .asd files, and asdf:load-op the *:com.informatimago.common-lisp* system, which depends on all the sublibrary systems.

If the variables are set correctly in *compile-with-asdf.lisp*, then the *compile.sh* script can be used to compile it with ccl, clisp, ecl and sbcl, which are the four implementations tested so far.

Compilation directly with ASDF

You can compile directly with ASDF, either by setting up symbolic links of the .asd files to a site-systems directory, or adding the directories where these .asd files lie to the *asdf:*central-registry*.*

It is advised to used asdf-binary-locations or ASDF-2 to have the objects placed in a good location.

Description of the sublibraries

Here is a short description of each sublibrary. See inside for details about the packages provided. The file system-graph.ps gives the dependencies between these sublibraries.

lisp-sexp/

com.informatimago.common-lisp.lisp-sexp.source-form

Functions to parse and manipulate Common Lisp sources as lisp forms (such as in macros).

lisp-reader/

com.informatimago.common-lisp.lisp-reader.reader

A Common Lisp reader, with some extensions (hooks to parse tokens).

lisp-text/

com.informatimago.common-lisp.lisp-text.source-text

This package exports functions to read and manipulate Common Lisp sources. Most of the text source properties are kept (file position, line number, comments, feature tests, etc), while no package is created and no symbol is interned.

lisp/

com.informatimago.common-lisp.lisp.generic-cl

This system provides various lisp packages or implementations.

generic-cl provides a package with the same symbols as COMMON-LISP, but whose functions are replaced by equivalent generic functions.

cesarum/

com.informatimago.common-lisp.cesarum.utility et al.

This sytem provies various utilities and extensions to Common Lisp. There are set of packages corresponding to Common Lisp chapters (package, list, string, array, etc), extending the Common Lisp types with useful functions. And there are additionnal abstract data types or data structures (dictionary, graph, left-leaning red-black trees).

file/

com.informatimago.common-lisp.file.file

Various utility to deal with files and streams.

arithmetic/

com.informatimago.common-lisp.aritmethic.primes

Various arithmetic utilities. For now, we have a package to compute primes (Eratostene sieve) and factorize integers.

data-encoding/

com.informatimago.common-lisp.data-encoding.data-encoding

A serializer/deserializer for lisp data.

heap/

A garbage collector for a heap of lisp data. (The purpose is to be used to share lisp data amongst different processes using shared memory).

csv/

com.informatimago.common-lisp.csv.csv

The CSV file format.

bank/

com.informatimago.common-lisp.bank.iban

A package to deal with IBAN (Internationnal Bank Account Numbers) and another to deal with RIB (Relevé d'Identité Bancaire) which are the French account numbers.

picture/

A ASCII-art picture package, with packages to draw lisp lists and trees.

rfc2822/

Utilities to deal with RFC-2822 message formats.

rfc3548/

This packages exports functions to encode an decode text blocks according to the encoding described in:

RFC3548: The Base16, Base32, and Base64 Data Encodings

unix/

Packages to read unix administration files /etc/group /etc/passwd and /etc/aliases.

html-base/

Some common HTML packages.

html-generator/

An HTML generator.

html-parser/

An HTML parser.

http/

Some utilities for HTTP processing.

diagram/

Some packages generating Diagram! 2.0 files (ancestor to OmniGrafle).

graphviz/

Generates GraphViz .dot files from com.informatimago.common-lisp.cesarum.graph objects.

interactive/

Interactive commands (to be used at the REPL).

invoice/

Process and generate invoices. (I don't use it anymore. There's a Devise datatype with a reader macro that should be extracted to be reused).

regexp/

An implementation of POSIX regexps and of EMACS regexps (both incomplete).

ed/

An implementation of ed(1) (lacks only regular expressions, to be provided eventually by the regexp sublibrary).

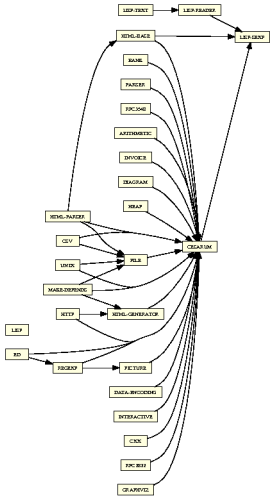
cxx/

Partial C++ parser and call graph builder.

make-depends/

Script to generate the dependencies (asd files) from the sources, and to generate summary.html files.

Dependencies of the sub-libraries



A dependency diagram of the com.informatimago.common-lisp sub-libraries