Common Lisp UltraSpec – A Project For Modern Common Lisp Documentation

Michał „phoe” Herda
`#lisp-pl` @ Freenode
Jagiellonian University, Cracow, Poland

Previously on
European Lisp Symposium 2016

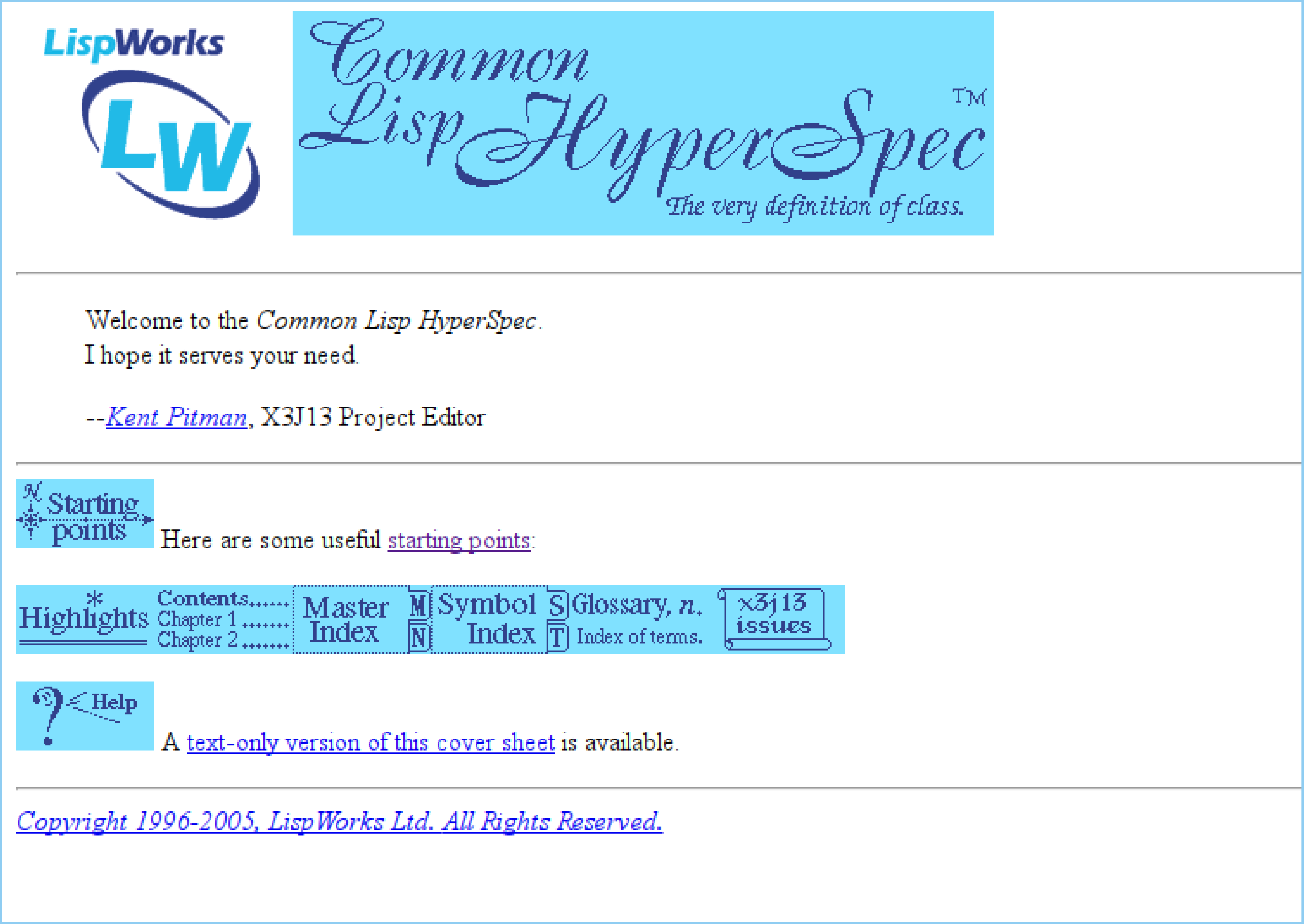
Yet Another Rant About The State Of Common Lisp Documentation

Michał „phoe” Herda

Presenting the
European Lisp Symposium 2017
Exclusive:

Yet Another Rant About The State Of Common Lisp Documentation, Part 2

Michał „phoe” Herda

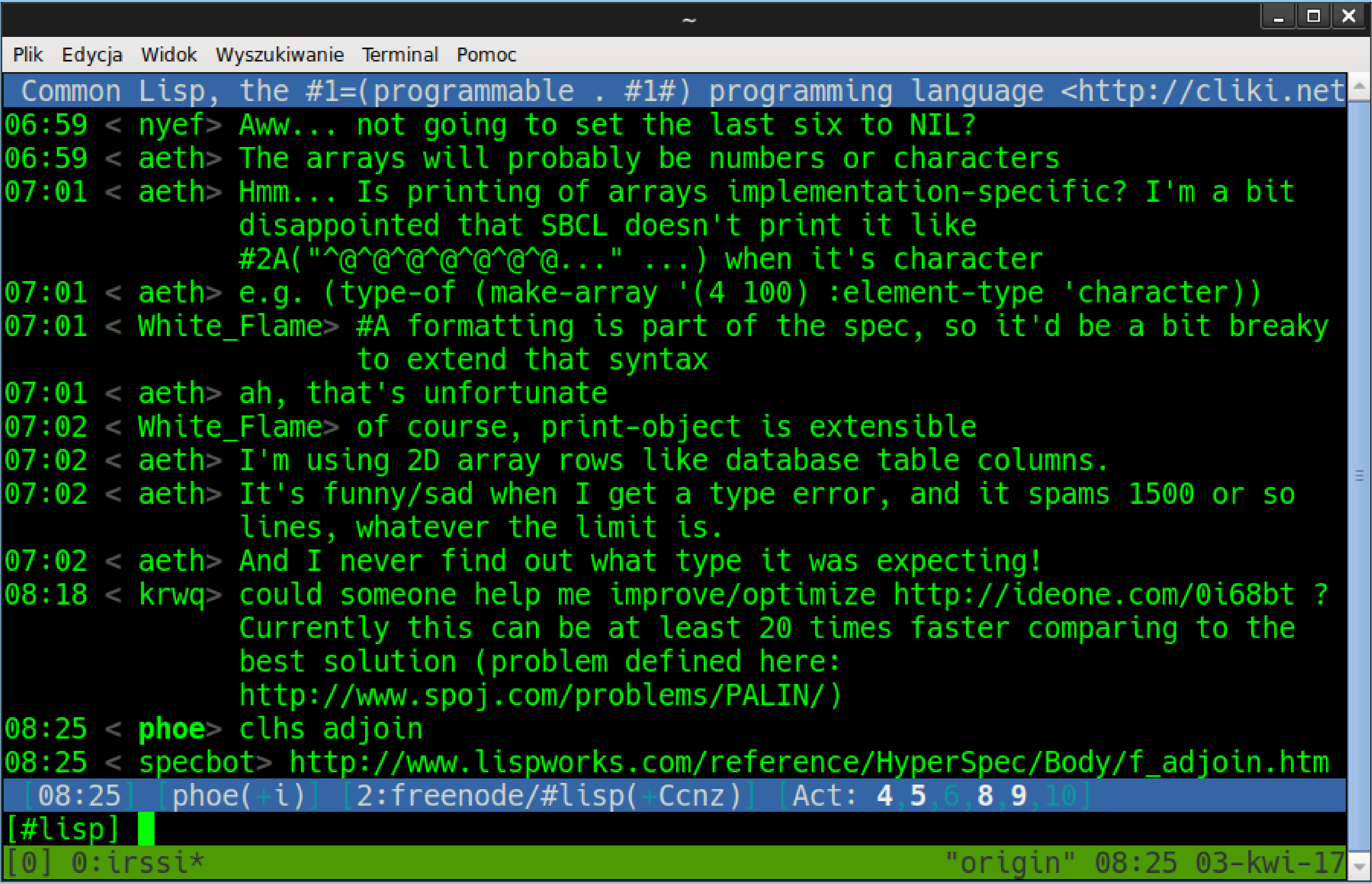


Welcome to the *Common Lisp HyperSpec*.
I hope it serves your need.

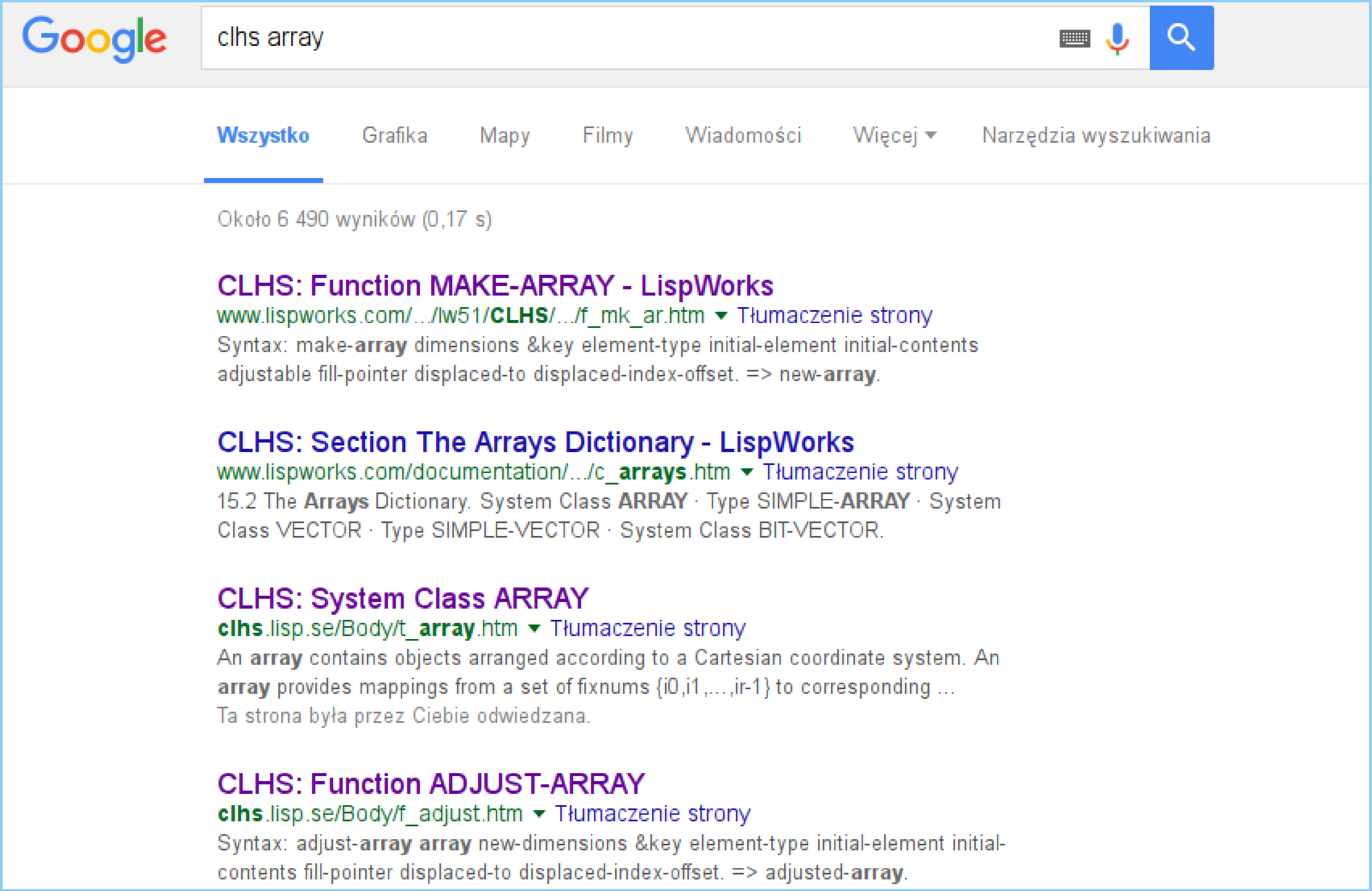
--*Kent Pitman*, X3J13 Project Editor

Here are some useful starting points:

A text-only version of this cover sheet is available.



```
 Common Lisp, the #1=(programmable . #1#) programming language <http://cliki.net
06:59 < nyef> Aww... not going to set the last six to NIL?
06:59 < aeth> The arrays will probably be numbers or characters
07:01 < aeth> Hmm... Is printing of arrays implementation-specific? I'm a bit
              disappointed that SBCL doesn't print it like
              #2A("^@^@^@^@^@^@^@..." ...) when it's character
07:01 < aeth> e.g. (type-of (make-array '(4 100) :element-type 'character))
07:01 < White_Flame> #A formatting is part of the spec, so it'd be a bit breaky
                     to extend that syntax
07:01 < aeth> ah, that's unfortunate
07:02 < White_Flame> of course, print-object is extensible
07:02 < aeth> I'm using 2D array rows like database table columns.
07:02 < aeth> It's funny/sad when I get a type error, and it spams 1500 or so
              lines, whatever the limit is.
07:02 < aeth> And I never find out what type it was expecting!
08:18 < krwq> could someone help me improve/optimize http://ideone.com/0i68bt ?
              Currently this can be at least 20 times faster comparing to the
              best solution (problem defined here:
              http://www.spoj.com/problems/PALIN/)
08:25 < phoe> clhs adjoin
08:25 < specbot> http://www.lispworks.com/reference/HyperSpec/Body/f_adjoin.htm
 [08:25] [phoe(+i)] [2:freenode/#lisp(+Ccnz)] [Act: 4,5,6,8,9,10]
[#lisp] 
[0] 0:irssi*                                         "origin" 08:25 03-kwi-17
```



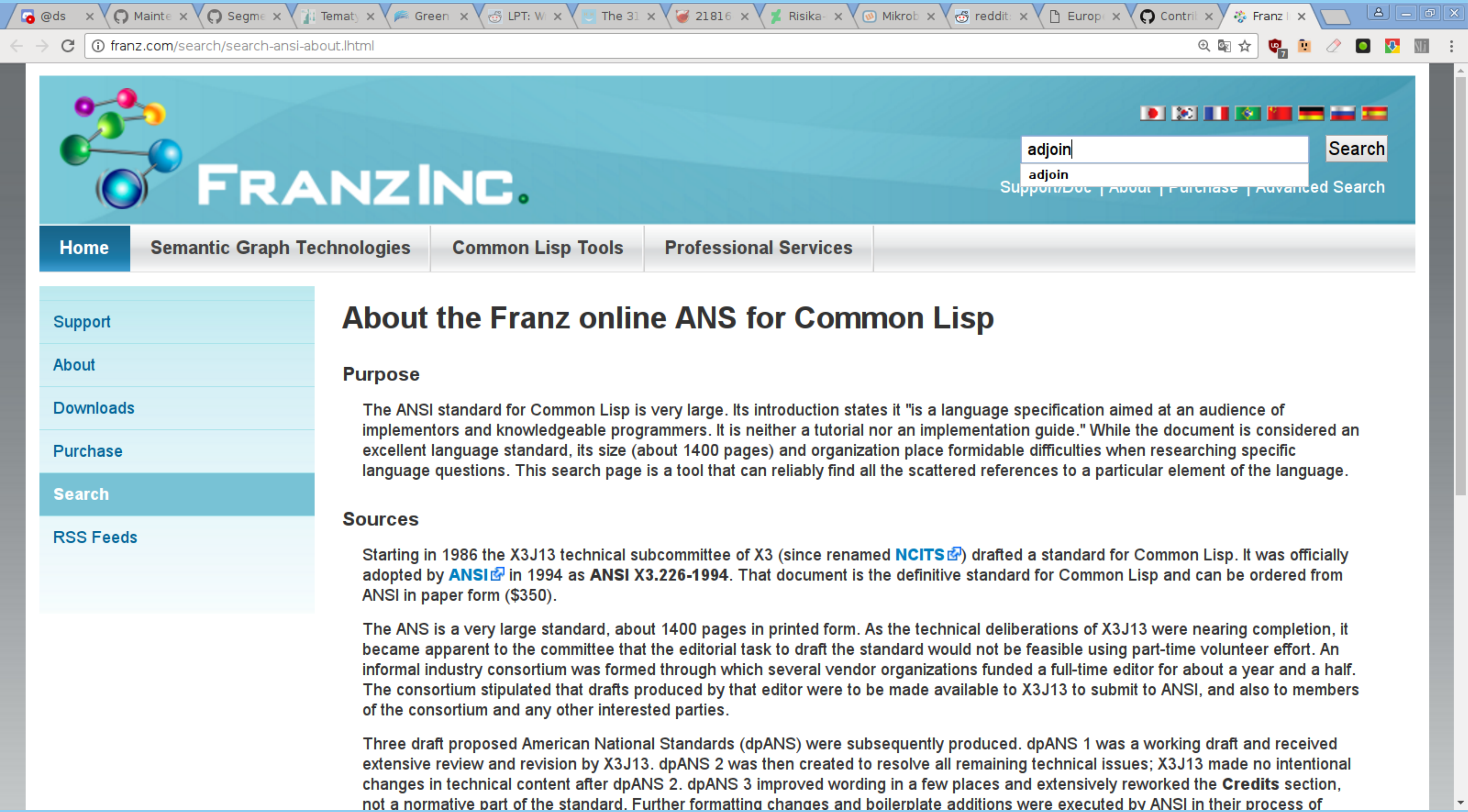
Około 6 490 wyników (0,17 s)

CLHS: Function MAKE-ARRAY - LispWorks
www.lispworks.com/.../lw51/CLHS/.../f_mk_ar.htm ▾ Tłumaczenie strony
Syntax: make-**array** dimensions &key element-type initial-element initial-contents adjustable fill-pointer displaced-to displaced-index-offset. => new-**array**.

CLHS: Section The Arrays Dictionary - LispWorks
www.lispworks.com/documentation/.../c_**arrays**.htm ▾ Tłumaczenie strony
15.2 The **Arrays** Dictionary. System Class **ARRAY** · Type SIMPLE-**ARRAY** · System Class VECTOR · Type SIMPLE-VECTOR · System Class BIT-VECTOR.

CLHS: System Class ARRAY
clhs.lisp.se/Body/t_**array**.htm ▾ Tłumaczenie strony
An **array** contains objects arranged according to a Cartesian coordinate system. An **array** provides mappings from a set of fixnums {i0,i1,...,ir-1} to corresponding ...
Ta strona była przez Ciebie odwiedzana.

CLHS: Function ADJUST-ARRAY
clhs.lisp.se/Body/f_adjust.htm ▾ Tłumaczenie strony
Syntax: adjust-**array** **array** new-dimensions &key element-type initial-element initial-contents fill-pointer displaced-to displaced-index-offset. => adjusted-**array**.


franz.com/search/search-ansi-about.lhtml
FRANZINC.
adjoin
Search
Support/Doc | About | Purchase | Advanced Search
Home
Semantic Graph Technologies
Common Lisp Tools
Professional Services
Support
About
Downloads
Purchase
Search
RSS Feeds

About the Franz online ANS for Common Lisp

Purpose

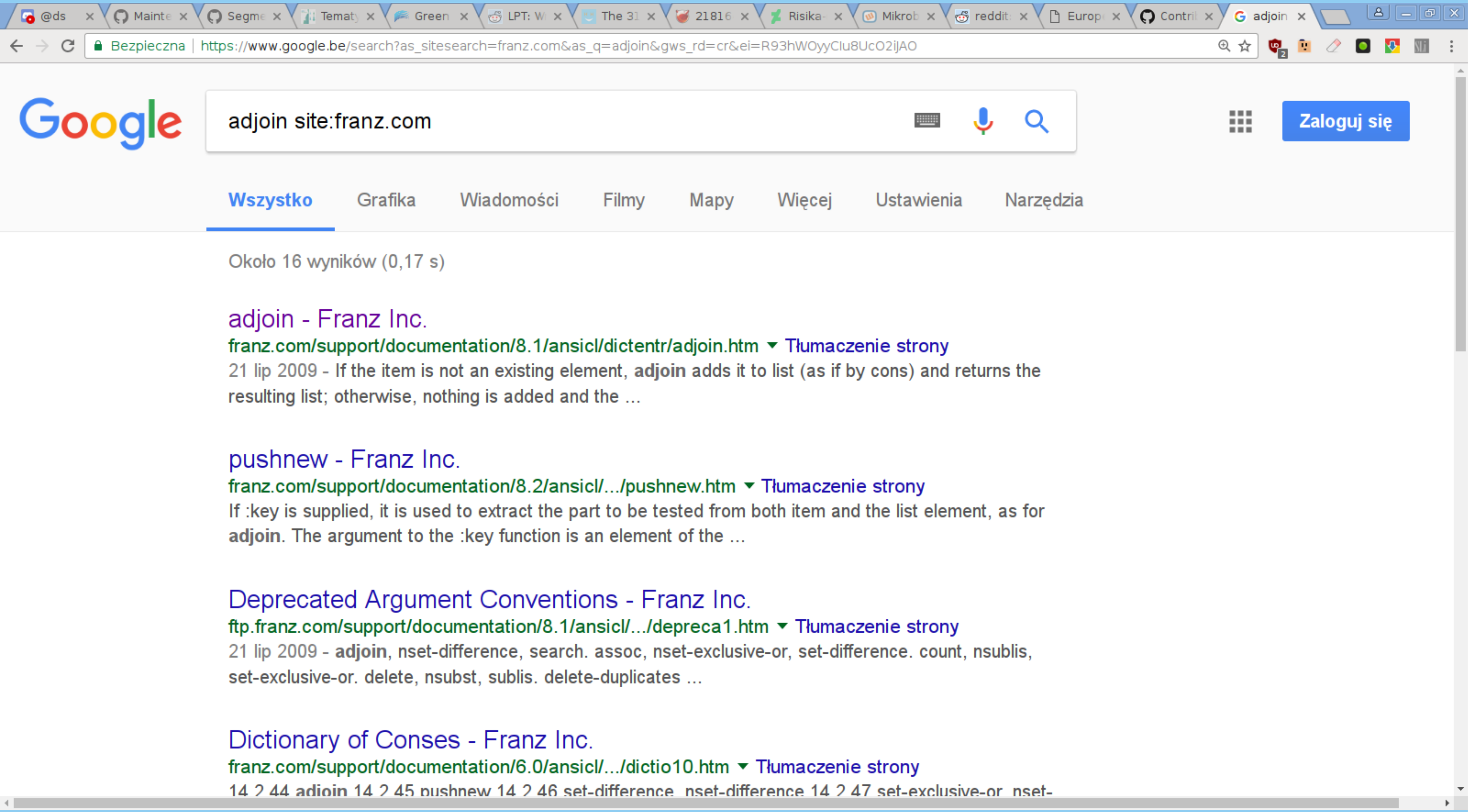
The ANSI standard for Common Lisp is very large. Its introduction states it "is a language specification aimed at an audience of implementors and knowledgeable programmers. It is neither a tutorial nor an implementation guide." While the document is considered an excellent language standard, its size (about 1400 pages) and organization place formidable difficulties when researching specific language questions. This search page is a tool that can reliably find all the scattered references to a particular element of the language.

Sources

Starting in 1986 the X3J13 technical subcommittee of X3 (since renamed **NCITS**) drafted a standard for Common Lisp. It was officially adopted by **ANSI** in 1994 as **ANSI X3.226-1994**. That document is the definitive standard for Common Lisp and can be ordered from ANSI in paper form ($350).

The ANS is a very large standard, about 1400 pages in printed form. As the technical deliberations of X3J13 were nearing completion, it became apparent to the committee that the editorial task to draft the standard would not be feasible using part-time volunteer effort. An informal industry consortium was formed through which several vendor organizations funded a full-time editor for about a year and a half. The consortium stipulated that drafts produced by that editor were to be made available to X3J13 to submit to ANSI, and also to members of the consortium and any other interested parties.

Three draft proposed American National Standards (dpANS) were subsequently produced. dpANS 1 was a working draft and received extensive review and revision by X3J13. dpANS 2 was then created to resolve all remaining technical issues; X3J13 made no intentional changes in technical content after dpANS 2. dpANS 3 improved wording in a few places and extensively reworked the **Credits** section, not a normative part of the standard. Further formatting changes and boilerplate additions were executed by ANSI in their process of



adjoin site:franz.com

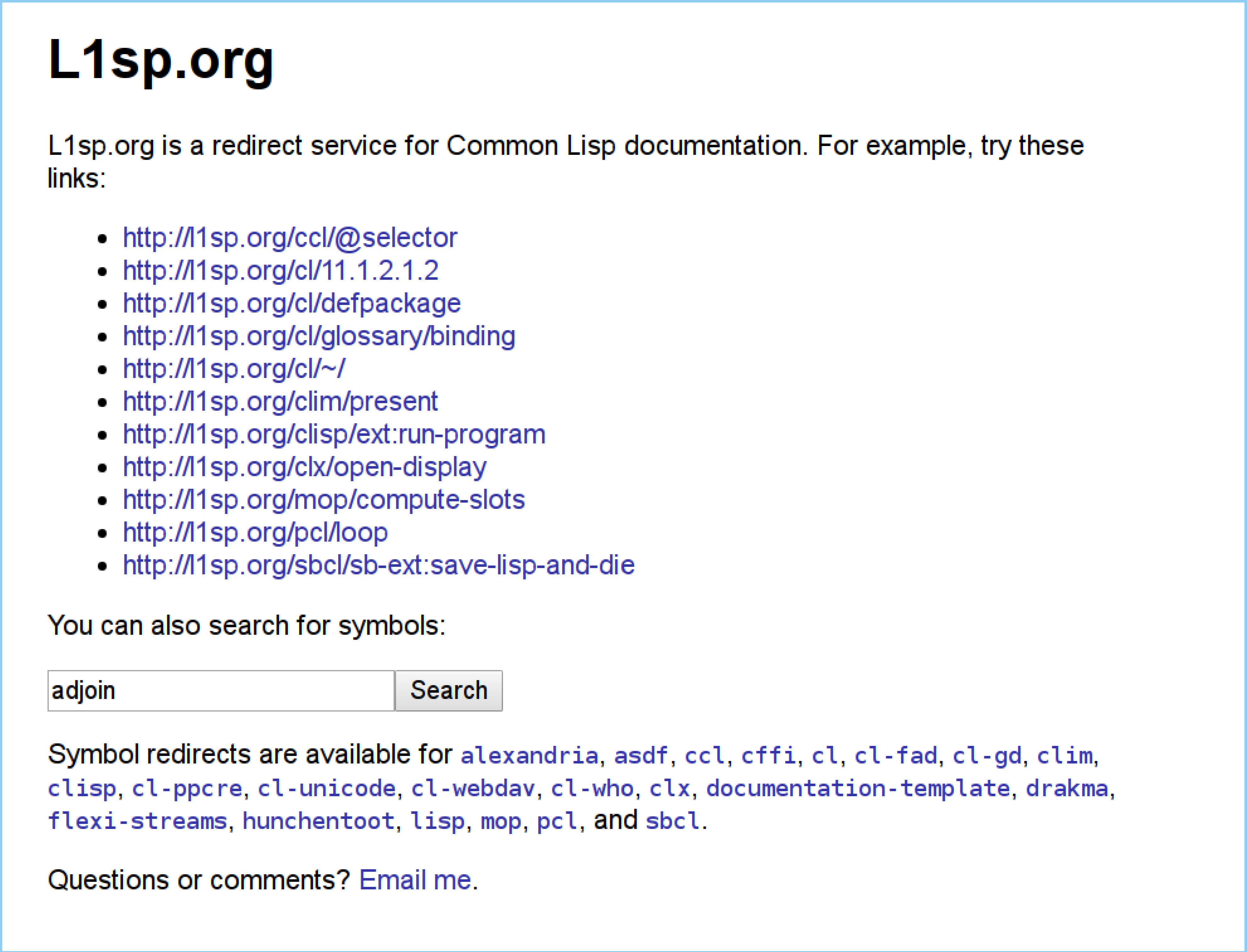
Około 16 wyników (0,17 s)

adjoin - Franz Inc.
franz.com/support/documentation/8.1/ansicl/dictentr/adjoin.htm ▾ Tłumaczenie strony
21 lip 2009 - If the item is not an existing element, **adjoin** adds it to list (as if by cons) and returns the resulting list; otherwise, nothing is added and the ...

pushnew - Franz Inc.
franz.com/support/documentation/8.2/ansicl/.../pushnew.htm ▾ Tłumaczenie strony
If :key is supplied, it is used to extract the part to be tested from both item and the list element, as for **adjoin**. The argument to the :key function is an element of the ...

Deprecated Argument Conventions - Franz Inc.
ftp.franz.com/support/documentation/8.1/ansicl/.../depreca1.htm ▾ Tłumaczenie strony
21 lip 2009 - **adjoin**, nset-difference, search. assoc, nset-exclusive-or, set-difference. count, nsublis, set-exclusive-or. delete, nsubst, sublis. delete-duplicates ...

Dictionary of Conses - Franz Inc.
franz.com/support/documentation/6.0/ansicl/.../dictio10.htm ▾ Tłumaczenie strony
14.2.44 **adjoin** 14.2.45 pushnew 14.2.46 set-difference, nset-difference 14.2.47 set-exclusive-or, nset-



L1sp.org

L1sp.org is a redirect service for Common Lisp documentation. For example, try these links:

- http://l1sp.org/ccl/@selector
- http://l1sp.org/cl/11.1.2.1.2
- http://l1sp.org/cl/defpackage
- http://l1sp.org/cl/glossary/binding
- http://l1sp.org/cl/~/
- http://l1sp.org/clim/present
- http://l1sp.org/clisp/ext:run-program
- http://l1sp.org/clx/open-display
- http://l1sp.org/mop/compute-slots
- http://l1sp.org/pcl/loop
- http://l1sp.org/sbcl/sb-ext:save-lisp-and-die

You can also search for symbols:

Symbol redirects are available for `alexandria`, `asdf`, `ccl`, `cffi`, `cl`, `cl-fad`, `cl-gd`, `clim`, `clisp`, `cl-ppcre`, `cl-unicode`, `cl-webdav`, `cl-who`, `clx`, `documentation-template`, `drakma`, `flexi-streams`, `hunchentoot`, `lisp`, `mop`, `pcl`, and `sbcl`.

Questions or comments? Email me.

```
http://www.lispworks.com/documentation/HyperSpec/...
http://www.sbcl.org/manual/...
http://ccl.clozure.com/manual/...
http://www.clisp.org/...
http://bauhh.dyndns.org:8000/clim-spec/...
http://bauhh.de/clxman/...
http://metamodular.com/CLOS-MOP/...

50+(?) more websites with library-specific docs
```

Quicklisp beta

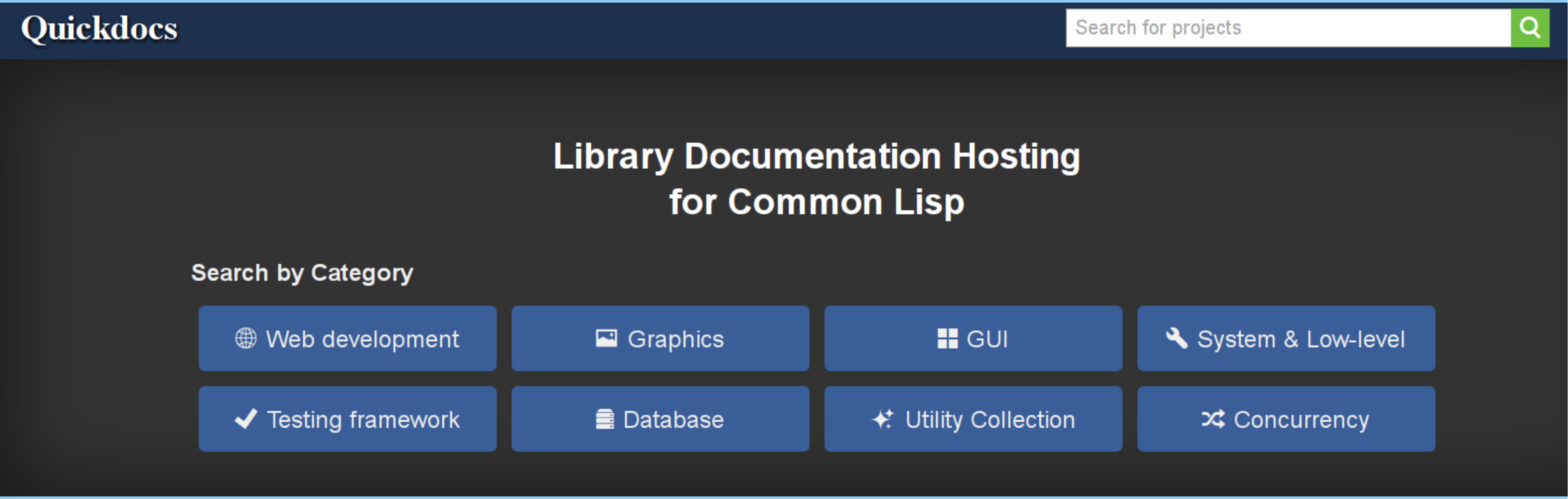
Quicklisp is a library manager for Common Lisp. It works with your existing Common Lisp implementation to download, install, and load any of over 1,400 libraries with a few simple commands.

Quicklisp is easy to install and works with ABCL, Allegro CL, Clasp, Clozure CL, CLISP, CMUCL, ECL, LispWorks, MKCL, SBCL, and Scieneer CL, on Linux, Mac OS X, and Windows. The libraries were last updated on February 27, 2017.

To get started with the Quicklisp beta, download and load
https://beta.quicklisp.org/quicklisp.lisp

PGP signature of quicklisp.lisp
sha256 of quicklisp.lisp =
4a7a5c2aebe0716417047854267397e24a44d0cce096127411e9ce9ccfeb2c17

Quicklisp is provided as-is without warranty of any kind. See the release notes for known problems.

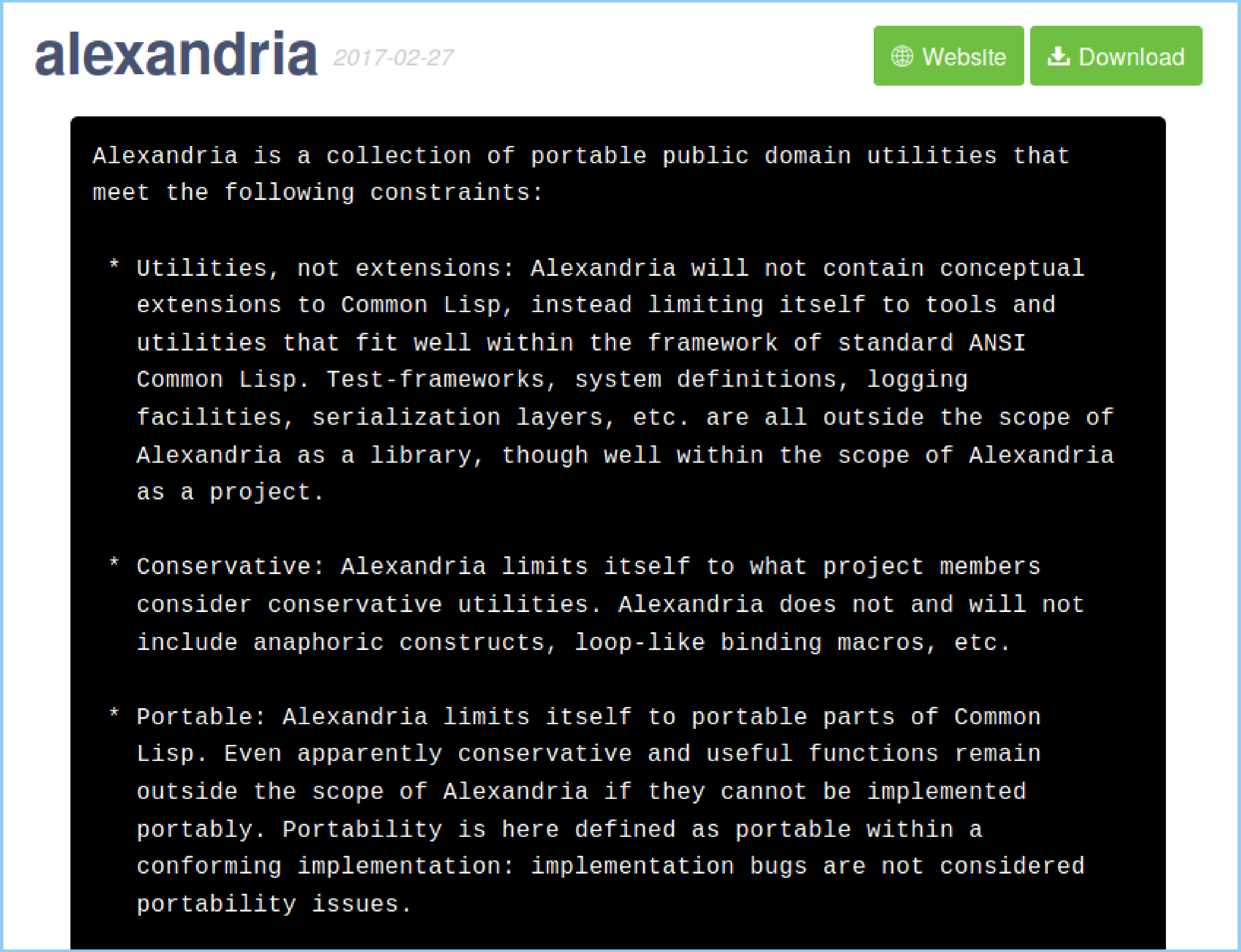

Quickdocs

Search for projects

Library Documentation Hosting
for Common Lisp

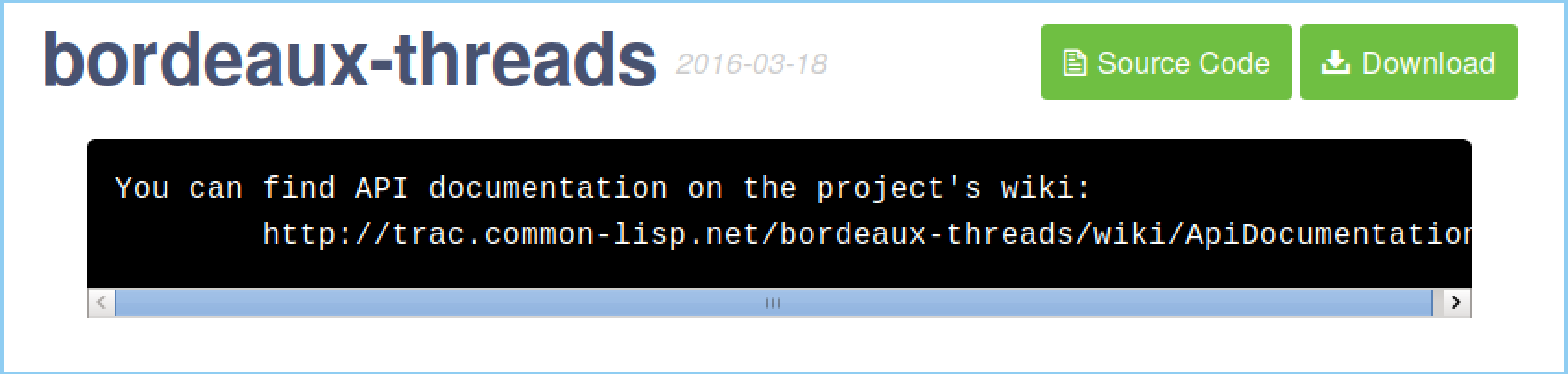
Search by Category

Web development
Graphics
GUI
System & Low-level
Testing framework
Database
Utility Collection
Concurrency



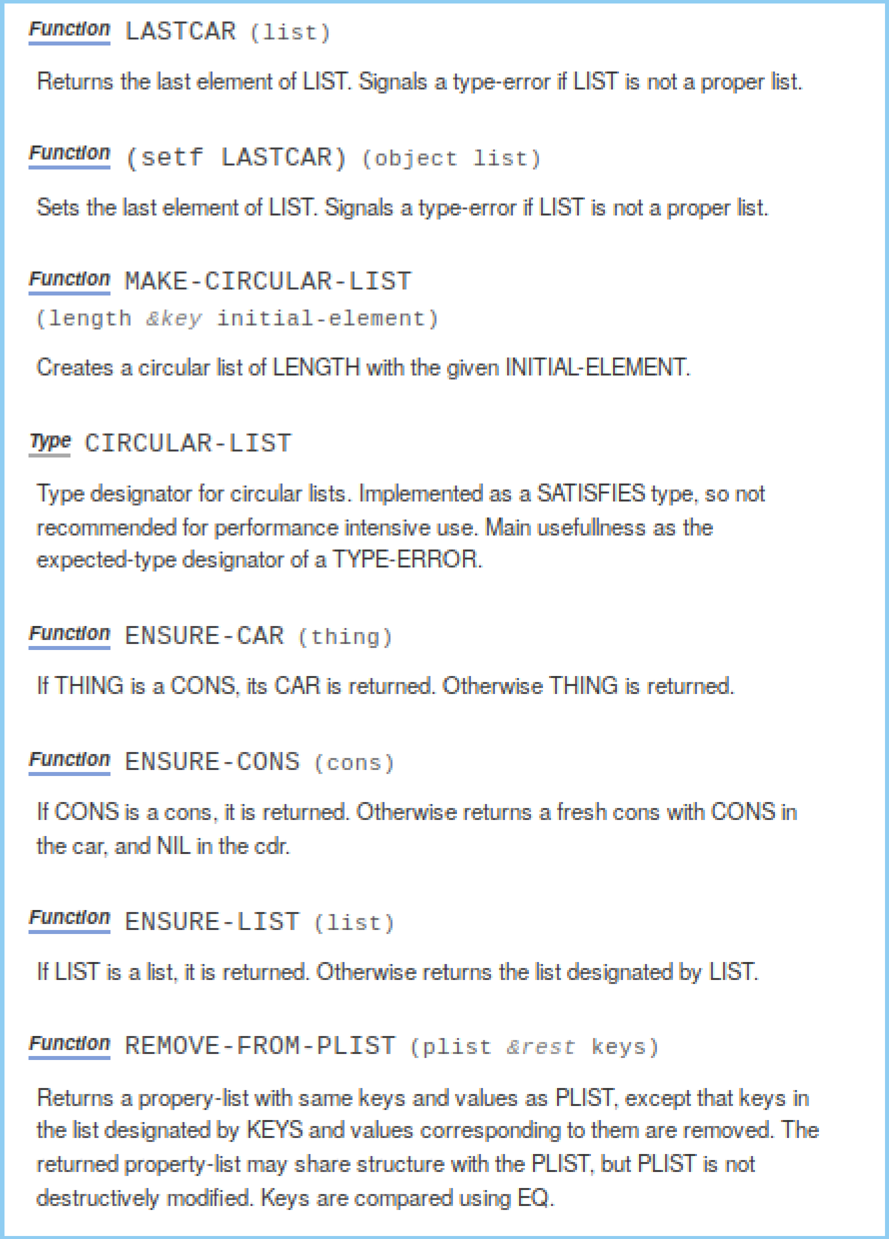
alexandria 2017-02-27

```
Alexandria is a collection of portable public domain utilities that
meet the following constraints:

 * Utilities, not extensions: Alexandria will not contain conceptual
   extensions to Common Lisp, instead limiting itself to tools and
   utilities that fit well within the framework of standard ANSI
   Common Lisp. Test-frameworks, system definitions, logging
   facilities, serialization layers, etc. are all outside the scope of
   Alexandria as a library, though well within the scope of Alexandria
   as a project.

 * Conservative: Alexandria limits itself to what project members
   consider conservative utilities. Alexandria does not and will not
   include anaphoric constructs, loop-like binding macros, etc.

 * Portable: Alexandria limits itself to portable parts of Common
   Lisp. Even apparently conservative and useful functions remain
   outside the scope of Alexandria if they cannot be implemented
   portably. Portability is here defined as portable within a
   conforming implementation: implementation bugs are not considered
   portability issues.
```



bordeaux-threads 2016-03-18

```
You can find API documentation on the project's wiki:
        http://trac.common-lisp.net/bordeaux-threads/wiki/ApiDocumentatio
```



Function `LASTCAR (list)`

Returns the last element of LIST. Signals a type-error if LIST is not a proper list.

Function `(setf LASTCAR) (object list)`

Sets the last element of LIST. Signals a type-error if LIST is not a proper list.

Function `MAKE-CIRCULAR-LIST (length &key initial-element)`

Creates a circular list of LENGTH with the given INITIAL-ELEMENT.

Type `CIRCULAR-LIST`

Type designator for circular lists. Implemented as a SATISFIES type, so not recommended for performance intensive use. Main usefullness as the expected-type designator of a TYPE-ERROR.

Function `ENSURE-CAR (thing)`

If THING is a CONS, its CAR is returned. Otherwise THING is returned.

Function `ENSURE-CONS (cons)`

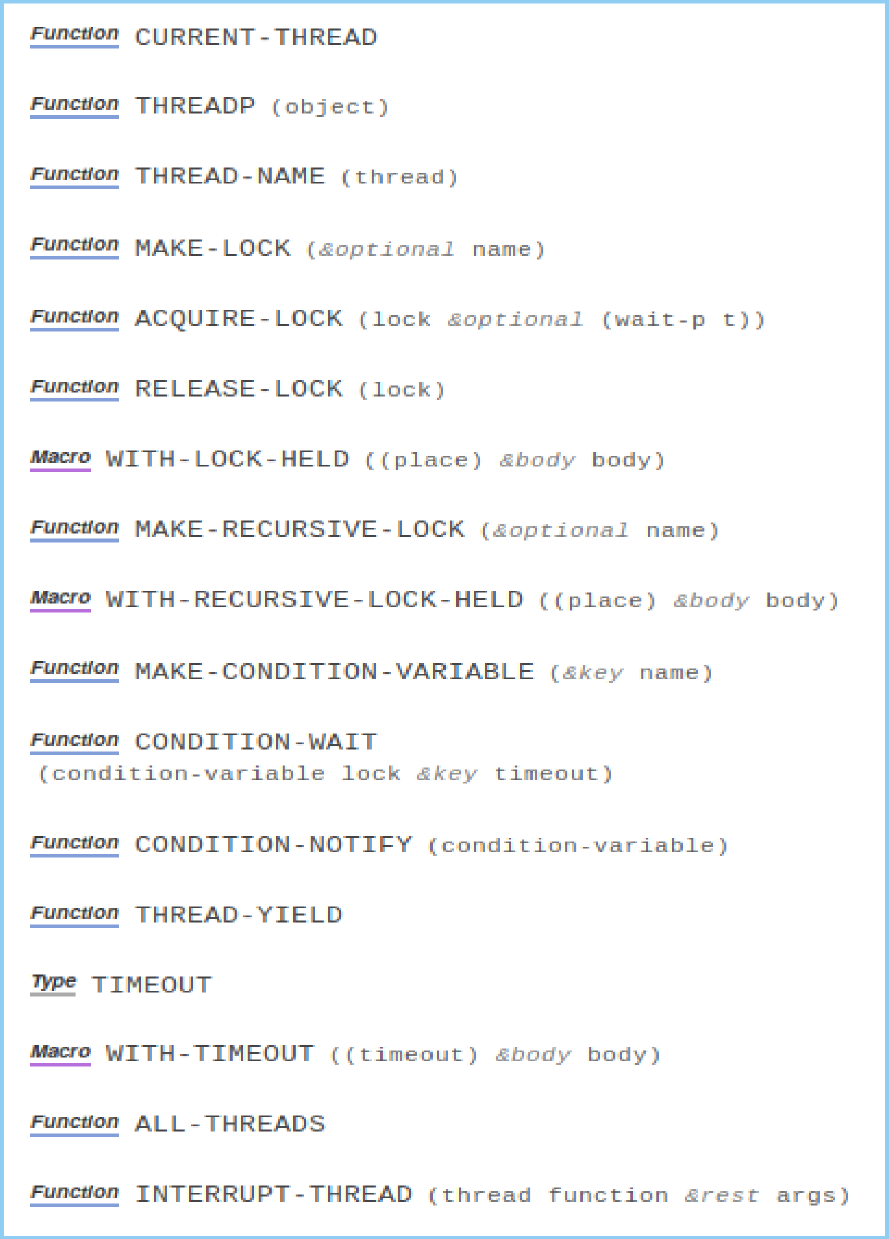
If CONS is a cons, it is returned. Otherwise returns a fresh cons with CONS in the car, and NIL in the cdr.

Function `ENSURE-LIST (list)`

If LIST is a list, it is returned. Otherwise returns the list designated by LIST.

Function `REMOVE-FROM-PLIST (plist &rest keys)`

Returns a propery-list with same keys and values as PLIST, except that keys in the list designated by KEYS and values corresponding to them are removed. The returned property-list may share structure with the PLIST, but PLIST is not destructively modified. Keys are compared using EQ.



Function `CURRENT-THREAD`

Function `THREADP (object)`

Function `THREAD-NAME (thread)`

Function `MAKE-LOCK (&optional name)`

Function `ACQUIRE-LOCK (lock &optional (wait-p t))`

Function `RELEASE-LOCK (lock)`

Macro `WITH-LOCK-HELD ((place) &body body)`

Function `MAKE-RECURSIVE-LOCK (&optional name)`

Macro `WITH-RECURSIVE-LOCK-HELD ((place) &body body)`

Function `MAKE-CONDITION-VARIABLE (&key name)`

Function `CONDITION-WAIT (condition-variable lock &key timeout)`

Function `CONDITION-NOTIFY (condition-variable)`

Function `THREAD-YIELD`

Type `TIMEOUT`

Macro `WITH-TIMEOUT ((timeout) &body body)`

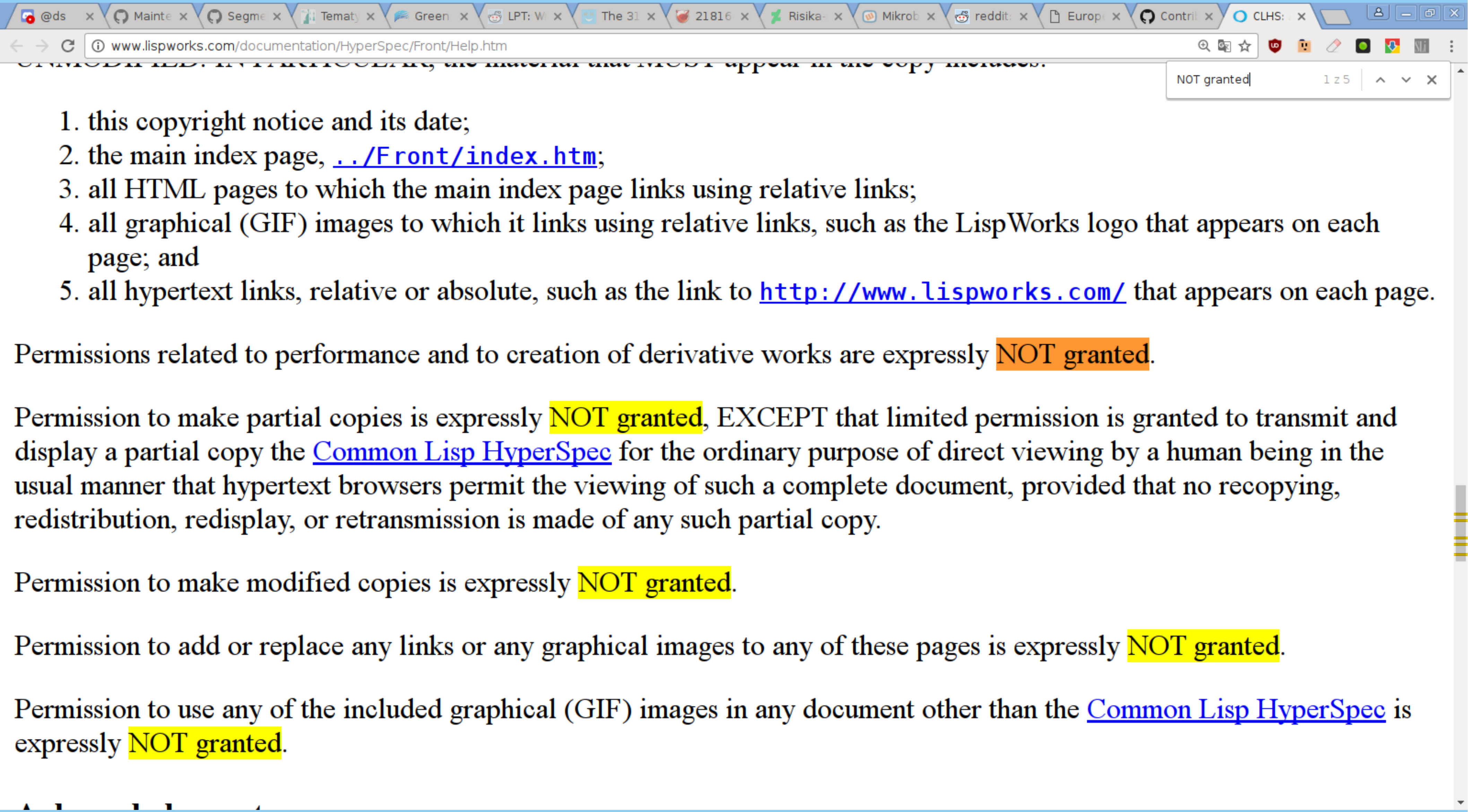
Function `ALL-THREADS`

Function `INTERRUPT-THREAD (thread function &rest args)`

Authorship Information

The Common Lisp HyperSpec is *not* the ANSI Common Lisp standard, but is based on that standard (with permission from ANSI and X3).

Content



1. this copyright notice and its date;
2. the main index page, ../Front/index.htm;
3. all HTML pages to which the main index page links using relative links;
4. all graphical (GIF) images to which it links using relative links, such as the LispWorks logo that appears on each page; and
5. all hypertext links, relative or absolute, such as the link to http://www.lispworks.com/ that appears on each page.

Permissions related to performance and to creation of derivative works are expressly NOT granted.

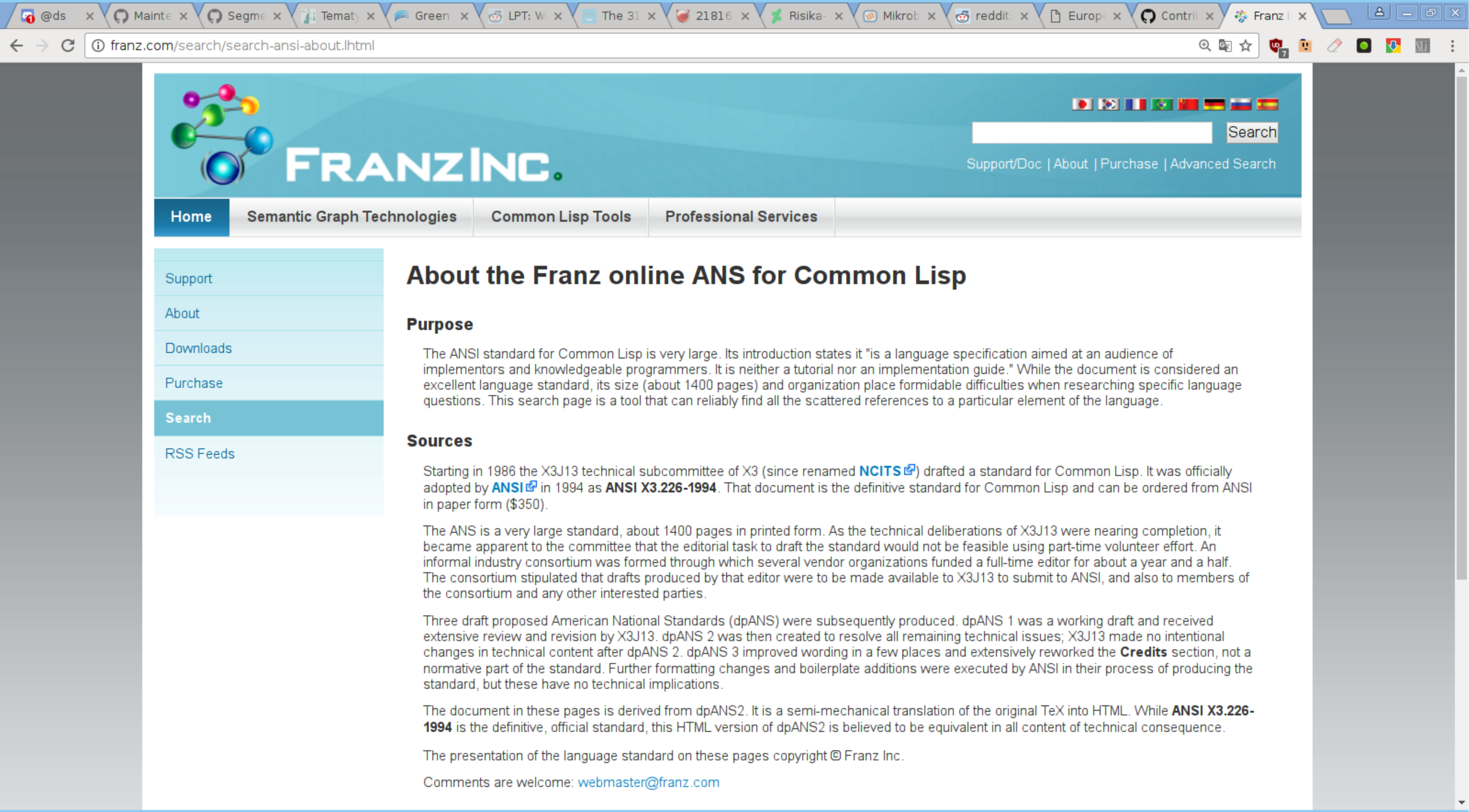
Permission to make partial copies is expressly NOT granted, EXCEPT that limited permission is granted to transmit and display a partial copy the Common Lisp HyperSpec for the ordinary purpose of direct viewing by a human being in the usual manner that hypertext browsers permit the viewing of such a complete document, provided that no recopying, redistribution, redisplay, or retransmission is made of any such partial copy.

Permission to make modified copies is expressly NOT granted.

Permission to add or replace any links or any graphical images to any of these pages is expressly NOT granted.

Permission to use any of the included graphical (GIF) images in any document other than the Common Lisp HyperSpec is expressly NOT granted.

Fin, Part 2
(...except it’s not)


franz.com/search/search-ansi-about.lhtml

FRANZ INC.

Search

Support/Doc | About | Purchase | Advanced Search

Home | Semantic Graph Technologies | Common Lisp Tools | Professional Services

- Support
- About
- Downloads
- Purchase
- Search
- RSS Feeds

About the Franz online ANS for Common Lisp

Purpose

The ANSI standard for Common Lisp is very large. Its introduction states it "is a language specification aimed at an audience of implementors and knowledgeable programmers. It is neither a tutorial nor an implementation guide." While the document is considered an excellent language standard, its size (about 1400 pages) and organization place formidable difficulties when researching specific language questions. This search page is a tool that can reliably find all the scattered references to a particular element of the language.

Sources

Starting in 1986 the X3J13 technical subcommittee of X3 (since renamed **NCITS**) drafted a standard for Common Lisp. It was officially adopted by **ANSI** in 1994 as **ANSI X3.226-1994**. That document is the definitive standard for Common Lisp and can be ordered from ANSI in paper form ($350).

The ANS is a very large standard, about 1400 pages in printed form. As the technical deliberations of X3J13 were nearing completion, it became apparent to the committee that the editorial task to draft the standard would not be feasible using part-time volunteer effort. An informal industry consortium was formed through which several vendor organizations funded a full-time editor for about a year and a half. The consortium stipulated that drafts produced by that editor were to be made available to X3J13 to submit to ANSI, and also to members of the consortium and any other interested parties.

Three draft proposed American National Standards (dpANS) were subsequently produced. dpANS 1 was a working draft and received extensive review and revision by X3J13. dpANS 2 was then created to resolve all remaining technical issues; X3J13 made no intentional changes in technical content after dpANS 2. dpANS 3 improved wording in a few places and extensively reworked the **Credits** section, not a normative part of the standard. Further formatting changes and boilerplate additions were executed by ANSI in their process of producing the standard, but these have no technical implications.

The document in these pages is derived from dpANS2. It is a semi-mechanical translation of the original TeX into HTML. While **ANSI X3.226-1994** is the definitive, official standard, this HTML version of dpANS2 is believed to be equivalent in all content of technical consequence.

The presentation of the language standard on these pages copyright © Franz Inc.

Comments are welcome: webmaster@franz.com

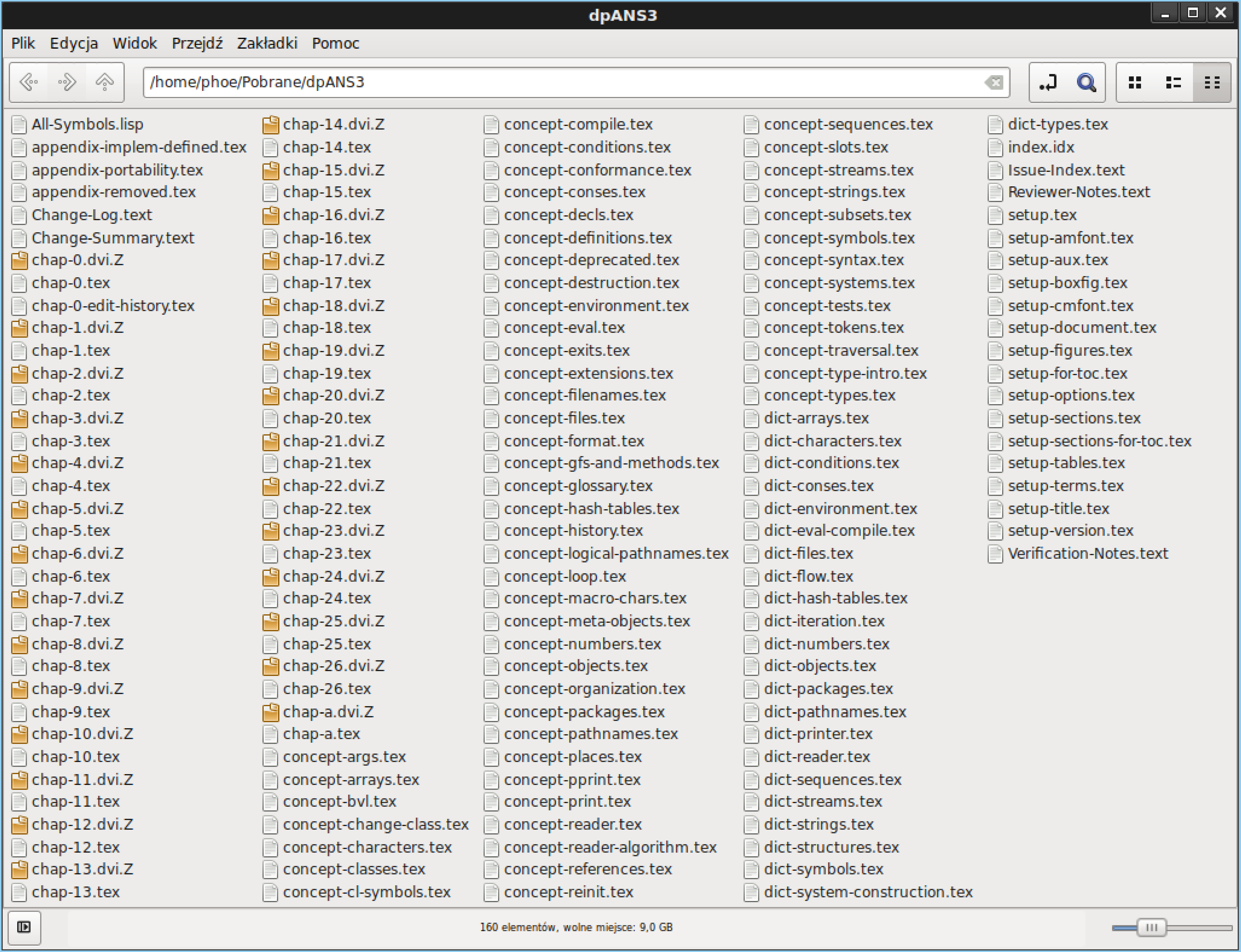
The document in these pages is derived from dpANS2. It is a semi-mechanical translation of the original TeX into HTML. While **ANSI X3.226-1994** is the definitive, official standard, this HTML version of dpANS2 is believed to be equivalent in all content of technical consequence.

The document in these pages is derived from dpANS2.

A Possible Solution To The State Of Common Lisp Documentation, Part 2

Common Lisp UltraSpec

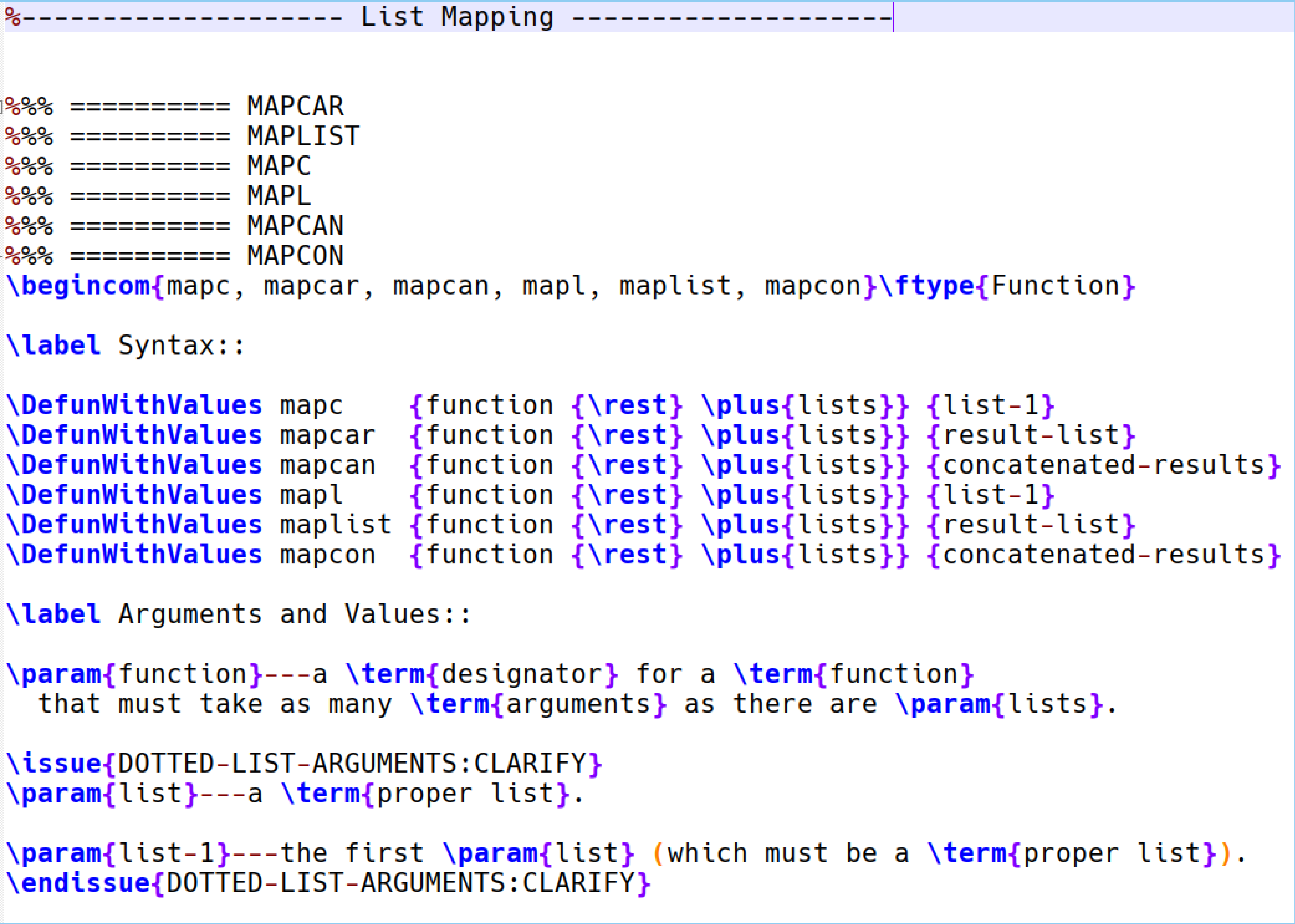
Michał „phoe” Herda



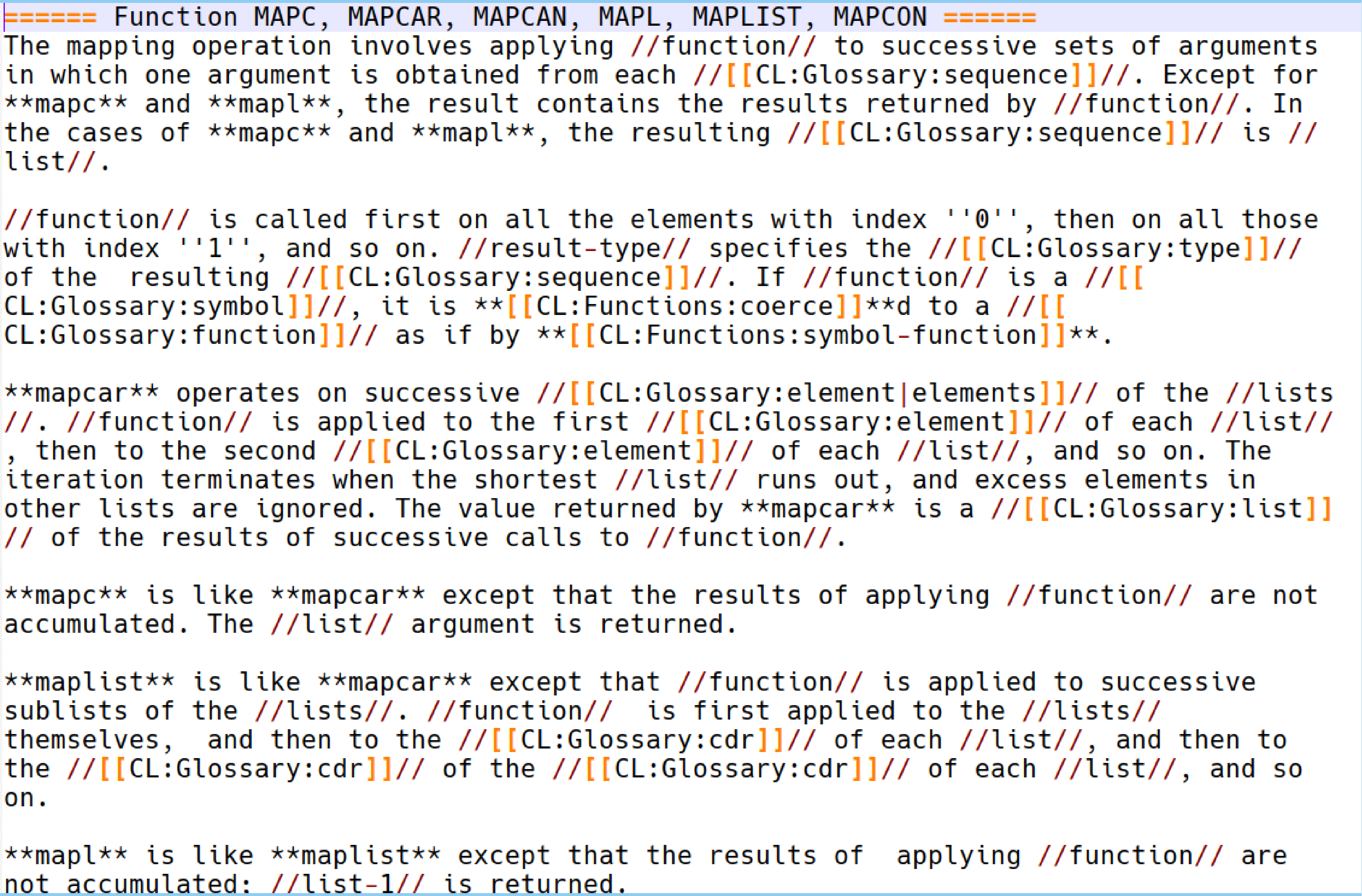
```
/home/phoe/Pobrane/dpANS3
```

```
All-Symbols.lisp
appendix-implem-defined.tex
appendix-portability.tex
appendix-removed.tex
Change-Log.text
Change-Summary.text
chap-0.dvi.Z
chap-0.tex
chap-0-edit-history.tex
chap-1.dvi.Z
chap-1.tex
chap-2.dvi.Z
chap-2.tex
chap-3.dvi.Z
chap-3.tex
chap-4.dvi.Z
chap-4.tex
chap-5.dvi.Z
chap-5.tex
chap-6.dvi.Z
chap-6.tex
chap-7.dvi.Z
chap-7.tex
chap-8.dvi.Z
chap-8.tex
chap-9.dvi.Z
chap-9.tex
chap-10.dvi.Z
chap-10.tex
chap-11.dvi.Z
chap-11.tex
chap-12.dvi.Z
chap-12.tex
chap-13.dvi.Z
chap-13.tex
chap-14.dvi.Z
chap-14.tex
chap-15.dvi.Z
chap-15.tex
chap-16.dvi.Z
chap-16.tex
chap-17.dvi.Z
chap-17.tex
chap-18.dvi.Z
chap-18.tex
chap-19.dvi.Z
chap-19.tex
chap-20.dvi.Z
chap-20.tex
chap-21.dvi.Z
chap-21.tex
chap-22.dvi.Z
chap-22.tex
chap-23.dvi.Z
chap-23.tex
chap-24.dvi.Z
chap-24.tex
chap-25.dvi.Z
chap-25.tex
chap-26.dvi.Z
chap-26.tex
chap-a.dvi.Z
chap-a.tex
concept-args.tex
concept-arrays.tex
concept-bvl.tex
concept-change-class.tex
concept-characters.tex
concept-classes.tex
concept-cl-symbols.tex
concept-compile.tex
concept-conditions.tex
concept-conformance.tex
concept-conses.tex
concept-decls.tex
concept-definitions.tex
concept-deprecated.tex
concept-destruction.tex
concept-environment.tex
concept-eval.tex
concept-exits.tex
concept-extensions.tex
concept-filenames.tex
concept-files.tex
concept-format.tex
concept-gfs-and-methods.tex
concept-glossary.tex
concept-hash-tables.tex
concept-history.tex
concept-logical-pathnames.tex
concept-loop.tex
concept-macro-chars.tex
concept-meta-objects.tex
concept-numbers.tex
concept-objects.tex
concept-organization.tex
concept-packages.tex
concept-pathnames.tex
concept-places.tex
concept-pprint.tex
concept-print.tex
concept-reader.tex
concept-reader-algorithm.tex
concept-references.tex
concept-reinit.tex
concept-sequences.tex
concept-slots.tex
concept-streams.tex
concept-strings.tex
concept-subsets.tex
concept-symbols.tex
concept-syntax.tex
concept-systems.tex
concept-tests.tex
concept-tokens.tex
concept-traversal.tex
concept-type-intro.tex
concept-types.tex
dict-arrays.tex
dict-characters.tex
dict-conditions.tex
dict-conses.tex
dict-environment.tex
dict-eval-compile.tex
dict-files.tex
dict-flow.tex
dict-hash-tables.tex
dict-iteration.tex
dict-numbers.tex
dict-objects.tex
dict-packages.tex
dict-pathnames.tex
dict-printer.tex
dict-reader.tex
dict-sequences.tex
dict-streams.tex
dict-strings.tex
dict-structures.tex
dict-symbols.tex
dict-system-construction.tex
dict-types.tex
index.idx
Issue-Index.text
Reviewer-Notes.text
setup.tex
setup-amfont.tex
setup-aux.tex
setup-boxfig.tex
setup-cmfont.tex
setup-document.tex
setup-figures.tex
setup-for-toc.tex
setup-options.tex
setup-sections.tex
setup-sections-for-toc.tex
setup-tables.tex
setup-terms.tex
setup-title.tex
setup-version.tex
Verification-Notes.text
```

160 elementów, wolne miejsce: 9,0 GB



```
%-------------------- List Mapping --------------------


%%% ========== MAPCAR
%%% ========== MAPLIST
%%% ========== MAPC
%%% ========== MAPL
%%% ========== MAPCAN
%%% ========== MAPCON
\begincom{mapc, mapcar, mapcan, mapl, maplist, mapcon}\ftype{Function}

\label Syntax::

\DefunWithValues mapc    {function {\rest} \plus{lists}} {list-1}
\DefunWithValues mapcar  {function {\rest} \plus{lists}} {result-list}
\DefunWithValues mapcan  {function {\rest} \plus{lists}} {concatenated-results}
\DefunWithValues mapl    {function {\rest} \plus{lists}} {list-1}
\DefunWithValues maplist {function {\rest} \plus{lists}} {result-list}
\DefunWithValues mapcon  {function {\rest} \plus{lists}} {concatenated-results}

\label Arguments and Values::

\param{function}---a \term{designator} for a \term{function}
 that must take as many \term{arguments} as there are \param{lists}.

\issue{DOTTED-LIST-ARGUMENTS:CLARIFY}
\param{list}---a \term{proper list}.

\param{list-1}---the first \param{list} (which must be a \term{proper list}).
\endissue{DOTTED-LIST-ARGUMENTS:CLARIFY}
```



```
====== Function MAPC, MAPCAR, MAPCAN, MAPL, MAPLIST, MAPCON ======
The mapping operation involves applying //function// to successive sets of arguments
in which one argument is obtained from each //[[CL:Glossary:sequence]]//. Except for
**mapc** and **mapl**, the result contains the results returned by //function//. In
the cases of **mapc** and **mapl**, the resulting //[[CL:Glossary:sequence]]// is //
list//.

//function// is called first on all the elements with index ''0'', then on all those
with index ''1'', and so on. //result-type// specifies the //[[CL:Glossary:type]]//
of the  resulting //[[CL:Glossary:sequence]]//. If //function// is a //[[
CL:Glossary:symbol]]//, it is **[[CL:Functions:coerce]]**d to a //[[
CL:Glossary:function]]// as if by **[[CL:Functions:symbol-function]]**.

**mapcar** operates on successive //[[CL:Glossary:element|elements]]// of the //lists
//. //function// is applied to the first //[[CL:Glossary:element]]// of each //list//
, then to the second //[[CL:Glossary:element]]// of each //list//, and so on. The
iteration terminates when the shortest //list// runs out, and excess elements in
other lists are ignored. The value returned by **mapcar** is a //[[CL:Glossary:list]]
// of the results of successive calls to //function//.

**mapc** is like **mapcar** except that the results of applying //function// are not
accumulated. The //list// argument is returned.

**maplist** is like **mapcar** except that //function// is applied to successive
sublists of the //lists//. //function//  is first applied to the //lists//
themselves,  and then to the //[[CL:Glossary:cdr]]// of each //list//, and then to
the //[[CL:Glossary:cdr]]// of the //[[CL:Glossary:cdr]]// of each //list//, and so
on.

**mapl** is like **maplist** except that the results of  applying //function// are
not accumulated: //list-1// is returned.
```

Function MAPC, MAPCAR, MAPCAN, MAPL, MAPLIST, MAPCON

Syntax

- **mapc** *function* `&rest` *lists+* → *list-1*
- **mapcar** *function* `&rest` *lists+* → *result-list*
- **mapcan** *function* `&rest` *lists+* → *concatenated-results*
- **mapl** *function* `&rest` *lists+* → *list-1*
- **maplist** *function* `&rest` *lists+* → *result-list*
- **mapcon** *function* `&rest` *lists+* → *concatenated-results*

Arguments and Values

- *function* - a *designator* for a *function* that must take as many *arguments* as there are *lists*.
- *list* - a *proper list*.
- *list-1* - the first *list* (which must be a *proper list*).
- *result-list* - a *list*.
- *concatenated-results* - a *list*.

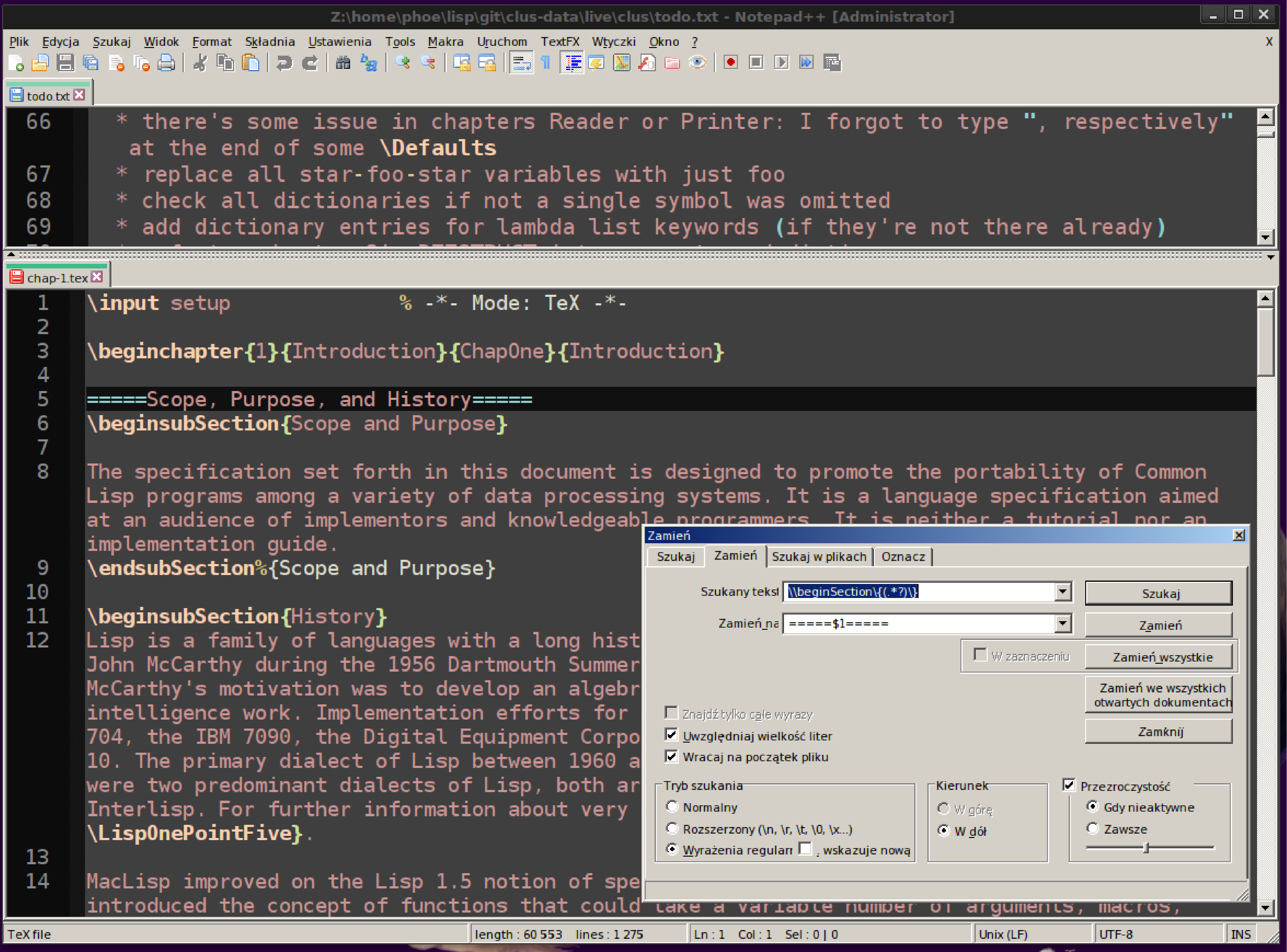
Description

The mapping operation involves applying *function* to successive sets of arguments in which one argument is obtained from each *sequence*. Except for **mapc** and **mapl**, the result contains the results returned by *function*. In the cases of **mapc** and **mapl**, the resulting *sequence* is *list*.

function is called first on all the elements with index `0`, then on all those with index `1`, and so on. *result-type* specifies the *type* of the resulting *sequence*. If *function* is a *symbol*, it is **coerce**d to a *function* as if by **symbol-function**.

mapcar operates on successive *elements* of the *lists*. *function* is applied to the first *element* of each *list*, then to the second *element* of each *list*, and so on. The iteration terminates when the shortest *list* runs out, and excess elements in other lists are ignored. The value returned by **mapcar** is a *list* of the results of successive calls to *function*.

mapc is like **mapcar** except that the results of applying *function* are not accumulated. The *list* argument is returned.


Z:\home\phoe\lisp\git\clus-data\live\clus\todo.txt - Notepad++ [Administrator]
Plik Edycja Szukaj Widok Format Składnia Ustawienia Tools Makra Uruchom TextFX Wtyczki Okno ?
todo.txt
chap-1.tex

```
 * there's some issue in chapters Reader or Printer: I forgot to type ", respectively"
   at the end of some \Defaults
 * replace all star-foo-star variables with just foo
 * check all dictionaries if not a single symbol was omitted
 * add dictionary entries for lambda list keywords (if they're not there already)
```

```
\input setup                % -*- Mode: TeX -*-

\beginchapter{1}{Introduction}{ChapOne}{Introduction}

=====Scope, Purpose, and History=====
\beginsubSection{Scope and Purpose}

The specification set forth in this document is designed to promote the portability of Common
Lisp programs among a variety of data processing systems. It is a language specification aimed
at an audience of implementors and knowledgeable programmers. It is neither a tutorial nor an
implementation guide.
\endsubSection%{Scope and Purpose}

\beginsubSection{History}
Lisp is a family of languages with a long hist
John McCarthy during the 1956 Dartmouth Summer
McCarthy's motivation was to develop an algebr
intelligence work. Implementation efforts for
704, the IBM 7090, the Digital Equipment Corpo
10. The primary dialect of Lisp between 1960 a
were two predominant dialects of Lisp, both ar
Interlisp. For further information about very
\LispOnePointFive}.

MacLisp improved on the Lisp 1.5 notion of spe
introduced the concept of functions that could take a variable number of arguments, macros,
```

Szukany tekst: \\beginSection\{(.*?)\}

Zamień na: =====$1=====

☐ W zaznaczeniu
☐ Znajdź tylko całe wyrazy
☑ Uwzględniaj wielkość liter
☑ Wracaj na początek pliku

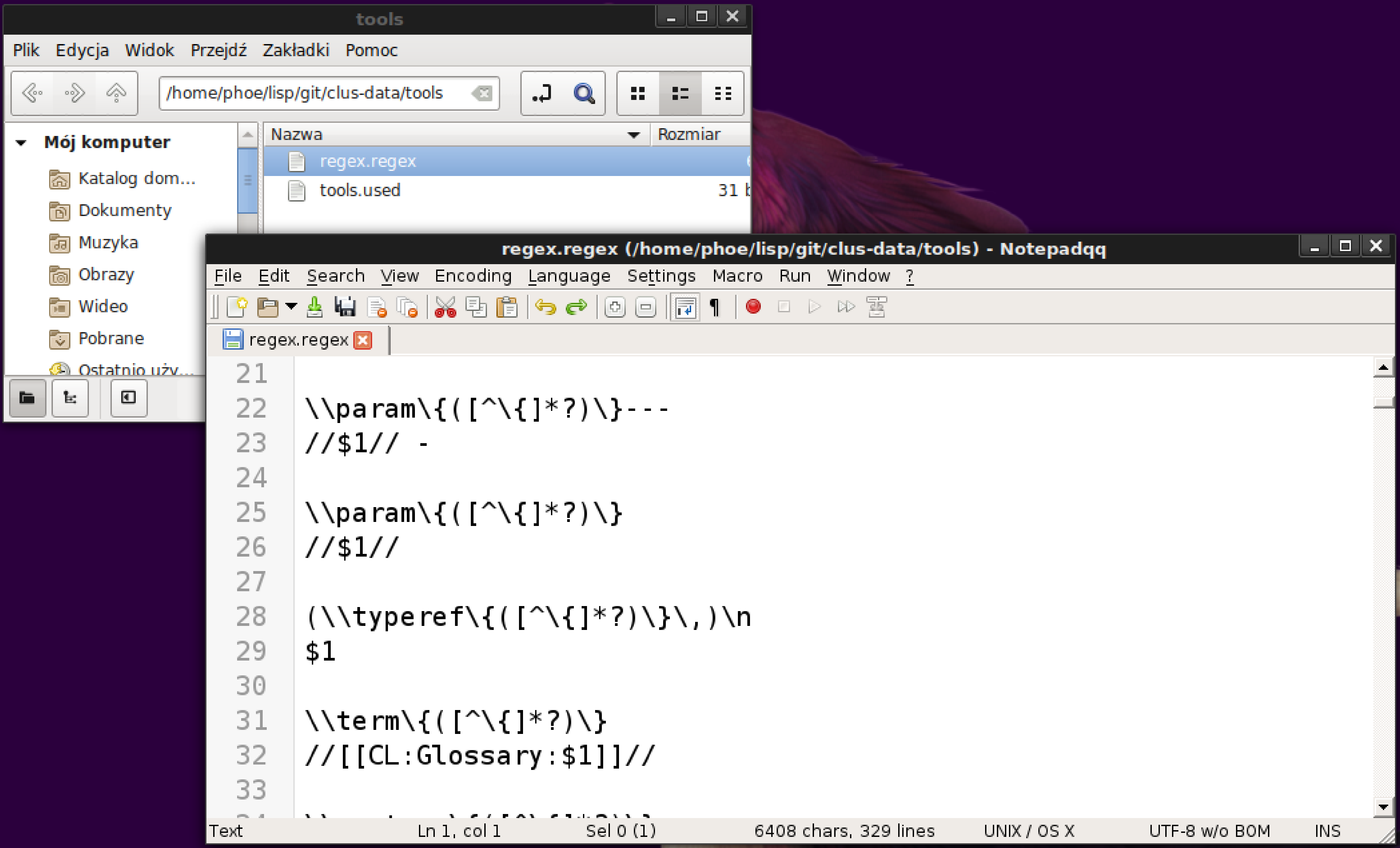
Tryb szukania: ○ Normalny ○ Rozszerzony (\n, \r, \t, \0, \x...) [✓ Wyrażenia regularne] ☐ . wskazuje nową

Kierunek: ○ W górę [✓ W dół]

☑ Przezroczystość: [✓ Gdy nieaktywne] ○ Zawsze

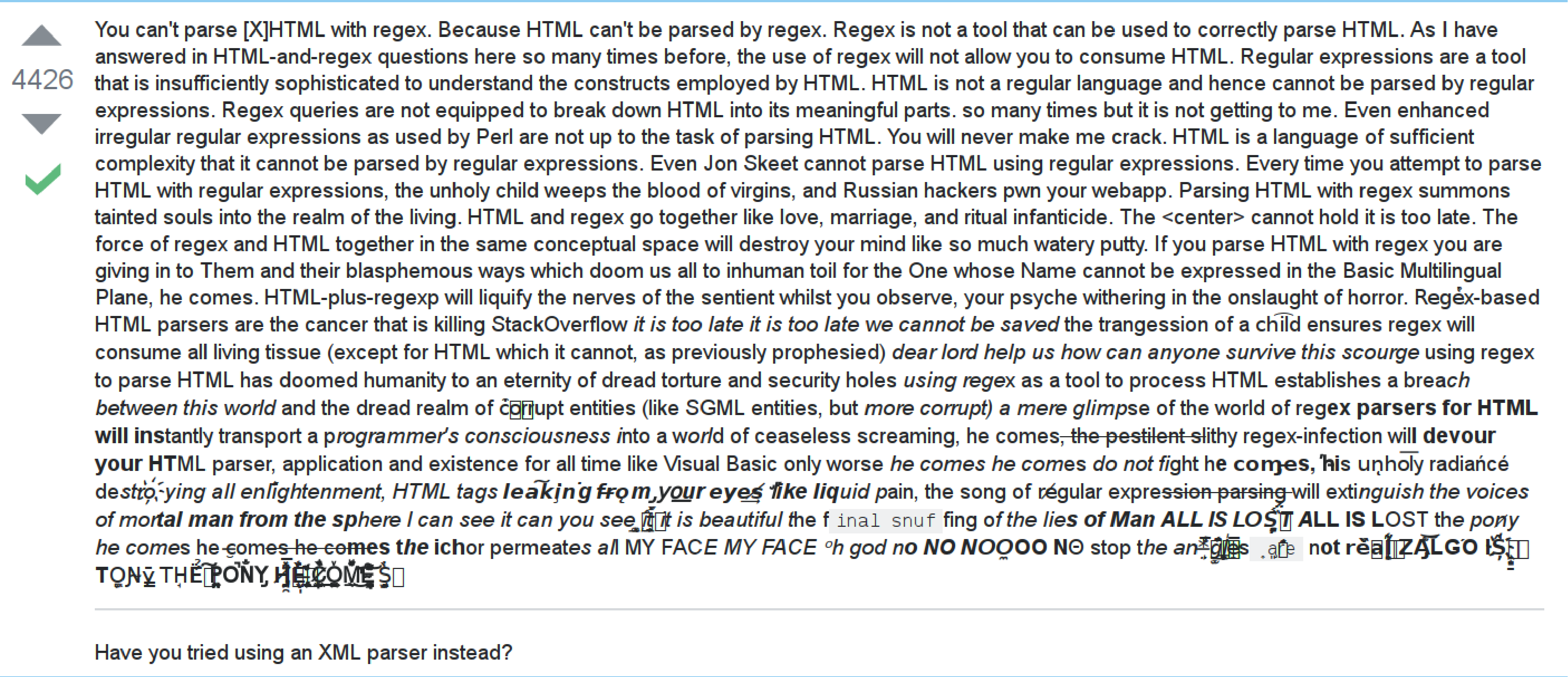
TeX file | length : 60 553 lines : 1 275 | Ln : 1 Col : 1 Sel : 0 | 0 | Unix (LF) | UTF-8 | INS

Parsing dpANS with RegEx


tools
Plik Edycja Widok Przejdź Zakładki Pomoc
/home/phoe/lisp/git/clus-data/tools
Mój komputer
Katalog dom...
Dokumenty
Muzyka
Obrazy
Wideo
Pobrane
Ostatnio uży...
Nazwa
Rozmiar
regex.regex
tools.used
31
regex.regex (/home/phoe/lisp/git/clus-data/tools) - Notepadqq
File Edit Search View Encoding Language Settings Macro Run Window ?
regex.regex
Text
Ln 1, col 1
Sel 0 (1)
6408 chars, 329 lines
UNIX / OS X
UTF-8 w/o BOM
INS

```

\\param\{([^\{]*?)\}---
//$1// -

\\param\{([^\{]*?)\}
//$1//

(\\typeref\{([^\{]*?)\}\,)\n
$1

\\term\{([^\{]*?)\}
//[[CL:Glossary:$1]]//

```

Parsing HTML (context-free grammar) with RegEx (regular grammar):



4426

✓

You can't parse [X]HTML with regex. Because HTML can't be parsed by regex. Regex is not a tool that can be used to correctly parse HTML. As I have answered in HTML-and-regex questions here so many times before, the use of regex will not allow you to consume HTML. Regular expressions are a tool that is insufficiently sophisticated to understand the constructs employed by HTML. HTML is not a regular language and hence cannot be parsed by regular expressions. Regex queries are not equipped to break down HTML into its meaningful parts. so many times but it is not getting to me. Even enhanced irregular regular expressions as used by Perl are not up to the task of parsing HTML. You will never make me crack. HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing HTML with regex summons tainted souls into the realm of the living. HTML and regex go together like love, marriage, and ritual infanticide. The <center> cannot hold it is too late. The force of regex and HTML together in the same conceptual space will destroy your mind like so much watery putty. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he comes. HTML-plus-regexp will liquify the nerves of the sentient whilst you observe, your psyche withering in the onslaught of horror. Rege̿x-based HTML parsers are the cancer that is killing StackOverflow *it is too late it is too late we cannot be saved* the trangession of a chi͡ld ensures regex will consume all living tissue (except for HTML which it cannot, as previously prophesied) *dear lord help us how can anyone survive this scourge* using regex to parse HTML has doomed humanity to an eternity of dread torture and security holes *using regex* as a tool to process HTML establishes a brea*ch between this world* and the dread realm of c͒ͪo͛ͫrrupt entities (like SGML entities, but *more corrupt) a mere glimp*se of the world of rege**x parsers for HTML will ins**tantly transport a p*rogrammer's consciousness i*nto a w*orl*d of ceaseless screaming, he comes~~, the pestilent s~~lithy regex-infection wil**l devour your HT**ML parser, application and existence for all time like Visual Basic only worse *he comes he comes do not fi*ght h**e com̡e̶s, ̕h̵i**s un̨ho͞ly radiańcé *destro҉ying all enli̍̈́̂̈́ghtenment, HTML tags* ***lea͠ki̧n͘g fr̶ǫm ̡yo͟ur eye͢s̸ ̛l̕ik͏e liq**uid* pain, the song of re̸gular expre~~ssion parsing~~ will exti*nguish the voices of mor*t***al man from the sp**here I can see it can you see ̲͚̖͔̙î̩́t̲͎̩̱͔́̋̀ it is beautiful t*he f `inal snuf` fing *o*f the lie*s of Man ALL IS LOŚ͖̩͇̗̪̏̈́T ALL I*S LOST th*e pon̷y he comes h*e ~~c̶̮omes he co~~**mes t*he** ich**or permeates al*l MY FACE MY FACE ᵒh god n*o NO NOO̼**OO N**Θ stop t*he an*̶͑̾̾̅ͫ͏̙̤g͇̫͛͆̾ͫ̑͆l͖͉̗̩̳̟̍ͫͥͨe̠̅s ` a̧͈͖r̽̾̈́͒͑e` n**ot rè̑ͧ̌aͨl̘̝̙̃ͤ͂̾̆ ZA̡͊͠͝LGΌ ISͮ̂҉̯͈͕̹̘̱ T**O͇̹̺ͅƝ̴ȳ̳ TH̘**Ë͖́̉ ͠P̯͍̭O̚N̐Y̡ H̸̡̪̯ͨ͊̽̅̾̎Ȩ̬̩̾͛ͪ̈́̀́͘ ̶̧̨̱̹̭̯ͧ̾ͬC̷̙̲̝͖ͭ̏ͥͮ͟Oͮ͏̮̪̝͍M̲̖͊̒ͪͩͬ̚̚͜Ȇ̴̟̟͙̞ͩ͌͝S̨̥̫͎̭ͯ̿̔̀ͅ**

Have you tried using an XML parser instead?

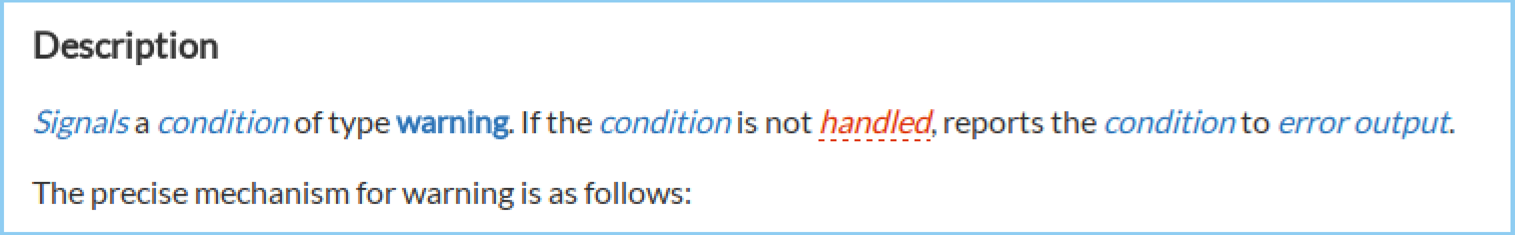
Parsing TeX (context-*sensitive* grammar) with RegEx (regular grammar):

Parsing a **known subset** of TeX **used in dpANS** with RegEx (regular grammar):

(with proper care)

What’s done so far?

- Done
  - All Dictionaries (e.g. **MAPCAR**, **ADJOIN**, …)
  - Glossary
- Not done / TODO
  - Concepts (e.g. **2.2 Reader Algorithm**)
  - Linking (esp. glossary entries)



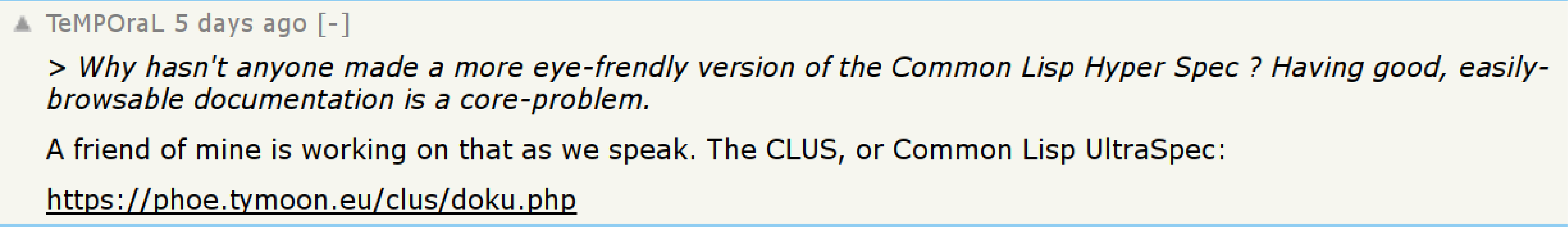
Description

Signals a *condition* of type **warning**. If the *condition* is not *handled*, reports the *condition* to *error output*.

The precise mechanism for warning is as follows:

What’s done so far?

- Done
  - All Dictionaries (e.g. **MAPCAR**, **ADJOIN**, …)
  - Glossary
- Not done / TODO
  - Concepts (e.g. **2.2 Reader Algorithm**)
  - Linking (esp. glossary entries)
  - Review
  - Diffs to the original dpANS
  - Once it’s finished, SHIP IT!



▲ TeMPOraL 5 days ago [-]

> *Why hasn't anyone made a more eye-frendly version of the Common Lisp Hyper Spec ? Having good, easily-browsable documentation is a core-problem.*

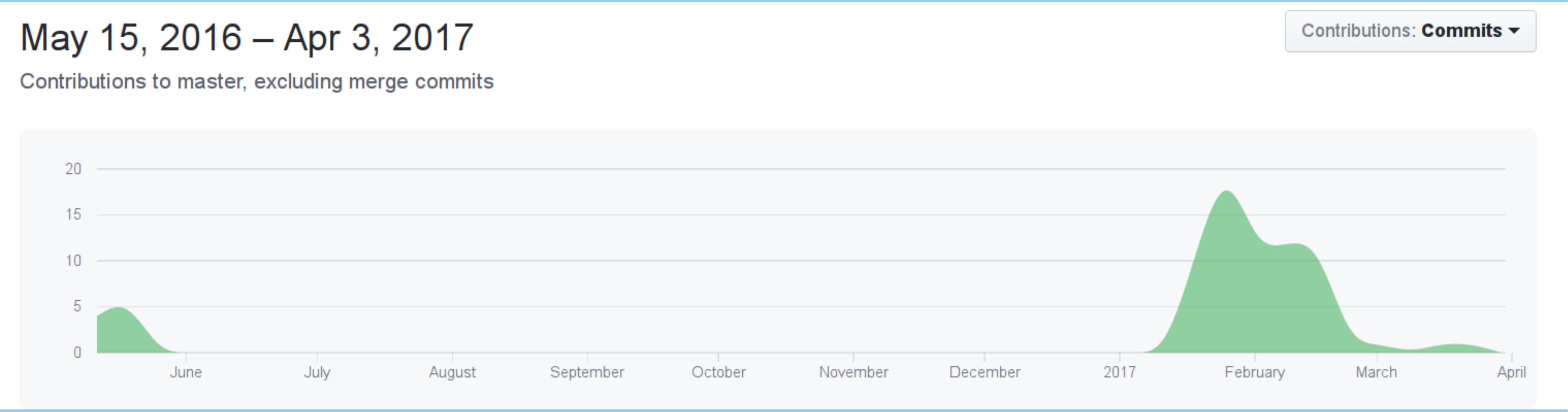
A friend of mine is working on that as we speak. The CLUS, or Common Lisp UltraSpec:

https://phoe.tymoon.eu/clus/doku.php

What am I aiming for?

- Editable
- Complete
- Downloadable
- Mirrorable/Clonable
- Versioned
- Modular
- Updatable
- Portable
- Unified
- Community-based

„Burning out”


May 15, 2016 – Apr 3, 2017

Contributions to master, excluding merge commits

ELS ‘16

...I think ELS is coming

WORK

ELS ‘17

Laziness ↑

...or is it?

Burnout ↑

...or is it?

Thoughts

- A translator, or an editor?
- A scribe, or a programmer?
- dpANS as humanistic material
- If I am a „new” editor of the specification, then who is the „new” X3J13?

Summing it up:

- Parse the dpANS specification
- Edit it where necessary
- Make it further parsable
- Give it to the community
- Let the community turn it into X
  - Emacs-readable version
  - Web version
  - Printable version
  - S-expression version
  - Community wiki
  - Repository for documents from the whole CL universe
  - Common Lisp the Language 3

Live demo/manifesto:

http://phoe.tymoon.eu/clus/

Thanks to Shinmera for the hosting!

The Actual End, Part 2
(...except it's not)

- because CLUS is not yet finished
- because I will be looking for contributors
- because it's questions time

Thanks for listening!