公開

002

TR-IT-0292

LISP から C への自動変換ツール (L2C) 説明書

西村 仁志 *　　隅田 英一郎
Hitoshi NISIMURA and Eiichiro SUMITA

1999 年 2 月 26 日

概 要

本書は LISP 言語で記述されたプログラムを等価な C 言語で記述されたプログラムに変換するシステム L2C (LISP to C) のマニュアルである。L2C は、Common Lisp の主要な関数やマクロをサポートしており、変換主導翻訳システムや文脈処理システムの変換の実績があり、また、他のシステムの変換にも活用できる。L2C は C 言語用リスト処理ライブラリ LinC と併用するように設計されている。

エイ・ティ・アール音声翻訳通信研究所
ATR Interpreting Telecommunications Research Laboratories

---

* (株) 国際電気通信基礎技術研究所

もくじ

表一覧

図一覧

第 1 章

システム概要

1.1 L2C の機能

Common Lisp で記述されたプログラムを、できるだけ人が読みやすいCプログラムに自動的に変換する。また、そのCプログラムを実行モジュールに変換する Makefile も出力する。

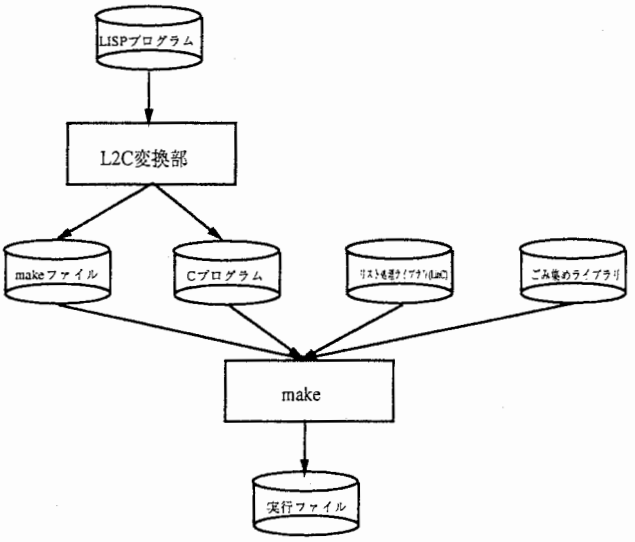

図 1.1: L2C のシステム構成

表 1.1: L2C ので利用するモジュールの説明

項目	内容
L2C 変換部	LISP プログラムを入力とし、それと同じ動きをする C プログラムを出力する
リスト処理ライブラリ（LinC）	本ライブラリは、L2C で変換されたプログラムが利用するものであり、Common Lisp で定義されている関数を C 言語で記述したものである。
ゴミ集めライブラリ	Xerox 社が PDS（パブリックドメイン）として提供している C 言語用のガベッジコレクションライブラリ

1.2 L2C の特徴

1. できるだけ、人が読みやすい C プログラムを出力する。
2. 一つのファイル名を指定するだけで、関連するすべてのファイルを変換することができる。
3. 変換後のCプログラムをコンパイル・リンクするための Makefile を自動的に作る。
4. 変換後のCプログラムは「リスト処理用ライブラリ (LinC)」を利用して記述されているので、修正が容易である。本ライブラリは、LISP のシステム関数と同機能のものをC言語で記述したライブラリである。
5. メモリ管理は、基本的にはゴミ集めライブラリを利用しているが、リスト処理用ライブラリで用意されているメモリ管理マクロを使うことにより、実行速度を上げることができる。

1.3 L2C の変換手順概要

L2C の変換手順の概要を以下に示す。

1. 変換対象の LISP プログラムのロードファイル名を指定する。
2. L2C がロードファイル名から関連するすべてのファイル名を取得する。
3. Cプログラムを実行モジュールに変換する Makefile も出力する。
4. Cプログラムに変換する。

1.4 L2C で扱えるデータ

入力プログラムの必要条件

- Common Lisp で記述されているプログラム

出力プログラムの稼働環境

- gcc、g++、Visual C++ でコンパイルリングできるCプログラム

第 2 章

利用方法

2.1 L2C の稼働環境の用意

L2C の稼働実績がある環境を表 2.1に示す。

表 2.1: L2C の稼働環境

機種	OS	LISP
SUN WorkStation	SunOS4.1.3	Allegro Common Lisp (4.3.1)
DEC Alpha Station	DigitalUnixV3.2	Allegro Common Lisp (4.3.1)
DOS/V	Linux	Allegro Common Lisp (4.3.1)

2.2 C プログラムの稼働実績環境の用意

C プログラムの稼働実績環境を表 2.2に示す。

表 2.2: C プログラムの稼働実績環境

機種	OS	コンパイラ	シェル
SUN WorkStation	SunOS4.1.3	gcc, g++	tcsh
DEC Alpha Station	DigitalUnixV3.2	gcc,g++	tcsh
KSR	KSROS	cc	tcsh
DOS/V	Linux	gcc, g++	tcsh
DOS/V	windows95	VisualC++	-

2.3 L2C をインストールする

1. 配布媒体から L2C と LinC と gc を、インストールするディレクトリにコピーする (gc, LinC, L2C の 3 つのディレクトリがコピーされる)

例： cp -r /export/* /home/.

2. ゴミ集めライブラリ（gc）をインストールする

   (a) ゴミ集めライブラリの READM(例:/home/gc/README) を見て、エディタで開き、実行ファイルを作る。

3. リスト処理ライブラリ（LinC）をインストールする。
   詳細は、[2] 参照のこと。

4. L2C 変換部（L2C）をインストールする

   (a) ファイル l2c-load.lisp エディタで開く

   (b) *L2C-dir* の値を、L2C をインストールしたパス名にする。
   例： (defvar *L2C-dir* "/home/L2C")

   (c) LinC-dir* の値を、LinC をインストールしたパス名にする。
   例： (defvar *LinC-dir* "/home/LinC")

   (d) リスプを起動する

   (e) L2C をロードする
   例： (load "/home/L2C/l2c-load")

   (f) L2C をコンパイルする
   例： (compile-l2c)

2.4 利用手順

1. Allegro Common Lisp を立ち上げる

2. インストールしたディレクトリに移る
   例： (cd "/home/L2C")

3. L2C をロードする
   例： (load "l2c-load")

4. ロードファイル名を指定し、システムに関係するファイル名を取得する
   例： (ana-file-all "ファイル名")
   結果は、*ana-file-list* にセットされる

5. LISP プログラムを C プログラムに変換し、同時に make ファイルも作成する
   ana-file-list* に あるファイルをすべて、変換する
   例： (lc-ana-file-all "/tmp/")

6. シェル環境で、変換結果であるCプログラムをmakeする
   例： make

2.5 エラーに対する原因とその対処

1. 変換時に指定されたログを見る。

2. 主なエラーに対する原因とその対処

   (a)
   - 現象： Cプログラムが全くできない。
   - 原因： リスププログラムのシンタックスエラー
   - 対処： 本L2Cは、シンタックスエラーがないLISPプログラムを前提にしています。そのため、リスププログラムのエラー箇所を修正したあと、変換して下さい。

   (b)
   - 現象： Cプログラム実行中、error_trapルーチンをcallする。
   - 原因： サポートされていないランタイム関数を呼び出したとき、error_trapルーチンをcallするようになっている。
   - 対処： L2Cでサポートしていない関数がファイルmain_main_ロードファイル名.c にまとめて定義してあるので、必要に応じて自分でその関数を記述する。

   (c)
   - 現象：Cプログラム実行中に異常終了する。
   - 原因：サポートされていないキーワード引数が変換対象のプログラムにある。
   - 対処：サポートされていないキーワード変数を、使わないようなLISPに変更してから変換しなおす。

   (d)
   - 現象：異常終了は、しないが結果がLISPと異なる。
   - 原因：コンバータの変換ミス
   - 対処： trace機能（『リスト処理用ライブラリ(LinC)』[2]の『利用方法』章の『デバッグ』節を参照）を利用し、エラーの原因を調べ、その箇所を手で修正する。

第 3 章

システム設計

3.1 変換方針

1. 変換結果であるCプログラムがLISPプログラムとの対応がつくものする
2. 人間が読むことができ、修正することができるCプログラムにする

3.2 変換方法

1. LISPの関数は、Cの関数に変換する
2. macroは、LISP側でマクロ展開したものをCに変換する

3.3 関数及び変数の変換規則

表3.1参照。

3.4 関数名の特殊変数を含む関数名の変換

表3.2参照。

表 3.1: 関数及び変数の変換規則

項目	LISP	C
LISP 関数	(car a)	car_o(a)
ユーザ定義関数	(foo a b)	foo(a, b)
グローバル変数	foo	FOO
ローカル変数	foo	foo
キーワード変数	:foo	KWD_FOO
重複する変数	(let ((a a))...)	a_1 = a;
パッケージ	a::b	A_b
-	a-b	a_b
*	*a*	ast_a_ast
.	a.b	a_dot_b
&	a&b	a_anpa_b
%	a%b	a_prcnt_b
?	a?b	a_question_b
!	a!b	a_exclamation_b
(	a(b	a_lparen_b
)	a)b	a_rparen_b
{	a{b	a_lmparen_b
}	a{b	a_rmparen_b
<	a	a_greater_b
>	a>b	a_less_b
+	a+b	a_plus_b
	a b	a_childer_b
/	a/b	a_slash_b
@	a@b	a_atto_b
#	a#b	a_sharp_b

表 3.2: 関数名の特殊変数を含む関数名の変換

LISP	C
1+	one_plus_o
1-	one_minus_o
+	plus_o
-	minus_o
*	times_o
/	divide_o
<	comp_greater_o
>	comp_less_o
<=	comp_greater_equal_o
>=	comp_less_equal_o
=	comp_equal_o
/=	comp_not_equal_o
string=	string_equal_o
string<	string_greater_o
char=	char_equal_o
char>	char_less_o

第 4 章

プログラム設計

4.1 モジュール構成

以下のように、2つのモジュールに分けることが出来る

1. 定義部 (l2c.lisp)
   変換可能関数および LISP と C 関数の名前の対応を定義する。
2. 変換部 (l2c-define.lisp)
   変換作業を行なう

4.2 変換アルゴリズム概要

以下に示すように、3パスで変換する。

1. 関連するファイル名の取得 (ana-file-all)
2. 全ての関数解析して関数名と引数を取得する (lc0-file-all)
3. 関数を変換する (lc-ana-file-all)

第 5 章

ファイル構成

5.1 L2C のファイル構成

表 5.1: L2C のファイル構成

内容	ファイル名	言語
システムのロード	l2c-load.lisp	Common Lisp
変換対象関数の定義	l2c-definitions.lisp	Common Lisp
変換部本体	l2c.lisp	Common Lisp

第 6 章

サポート状況

6.1 サポート状況概要

1. 出来るもの
   マクロ、パッケージ、lexical クロージャー、catch と throw、setf メソッド(ただし、一部のメソッドのみ)、ハッシュテーブル、format(ただし、一部記述子のみ)、read

2. 出来ないもの
   unwind-protect, 変数の shadowing, backquotes, bignums など

6.2 サポート状況サマリー

表 6.1: サポート状況サマリーテーブル

関数及びスペシャルフォーム	数
サポートしている	270
サポートしていない	471
全リスプ LISP(合計)	741

6.3 サポート状況の詳細

以下の記号で、示す。

1. サポートしているは、○

2. サポートしていない、×

3. キーワードを含む関数の場合は、サポートしているキーワードを表示している
   例：(find (a b &key (test (function eql)) (key (function identity))))

*	○
+	○
-	(- (nubmer1 number2))
/	(/ (nubmer1 number2))
/=	(/= (nubmer1 number2))
1+	(1+ (object))
1-	(1- (object))
<	○
<=	○
=	(= (nubmer1 number2))
>	○
>=	○
abort	×
abs	×
acons	(acons (key datum alist))
acos	×
acosh	×
add-method	×
adjoin	(adjoin (item list &key (test (function eql)) (key (function identity))))
adjust-array	×
adjustable-array-p	×
allocate-instance	×
alpha-char-p	(alpha-char-p (char))
alphanumericp	×
and	○
append	○
apply	○
apropos	×
apropos-list	×
aref	○

arithmetic-error-operands	×
arithmetic-error-operation	×
array-dimension	(array-dimension (array n))
array-dimensions	×
array-displacement	×
array-element-type	×
array-has-fill-pointer-p	×
array-in-bounds-p	×
array-rank	×
array-row-major-index	×
array-total-size	×
arrayp	(arrayp (object))
ash	(ash (value count))
asin	×
asinh	×
assert	×
assoc	(assoc (object a-list &key (test (function eql)) (key (function identity))))
assoc-if	×
assoc-if-not	×
atan	×
atanh	×
atom	(atom (object))
bit	×
bit-and	×
bit-andc1	×
bit-andc2	×
bit-eqv	×
bit-ior	×
bit-nand	×
bit-nor	×

bit-not	×
bit-orc1	×
bit-orc2	×
bit-vector-p	×
bit-xor	×
block	○
boole	×
both-case-p	×
boundp	×
break	×
broadcast-stream-streams	×
butlast	(butlast (object))
byte	×
byte-position	×
byte-size	×
caaaar	(caaaar (object))
caaadr	(caaadr (object))
caaar	(caaar (object))
caadar	(caadar (object))
caaddr	(caaddr (object))
caadr	(caadr (object))
caar	(caar (object))
cadaar	(cadaar (object))
cadadr	(cadadr (object))
cadar	(cadar (object))
caddar	(caddar (object))
cadddr	(cadddr (object))
caddr	(caddr (object))
cadr	(cadr (object))
car	(car (object))

case	○
catch	○
ccase	×
cdaaar	(cdaaar (object))
cdaadr	(cdaadr (object))
cdaar	(cdaar (object))
cdadar	(cdadar (object))
cdaddr	(cdaddr (object))
cdadr	(cdadr (object))
cdar	(cdar (object))
cddaar	(cddaar (object))
cddadr	(cddadr (object))
cddar	(cddar (object))
cdddar	(cdddar (object))
cddddr	(cddddr (object))
cdddr	(cdddr (object))
cddr	(cddr (object))
cdr	(cdr (object))
ceiling	×
cell-error-name	×
cerror	×
change-class	×
char	(char (char1 char2))
char-code	(char-code (char))
char-downcase	(char-downcase (char))
char-equal	○
char-greaterp	○
char-int	×
char-lessp	○
char-name	×

char-not-equal	○
char-not-greaterp	×
char-not-lessp	×
char-upcase	(char-upcase (char))
char/=	×
char<	○
char<=	○
char=	○
char>	○
char>=	○
character	(character (char))
characterp	(characterp (object))
check-type	×
cis	×
class-name	×
class-of	×
clear-input	×
clear-output	×
close	(close (stream))
clrhash	(clrhash (object))
code-char	(code-char (code &optional bit font))
coerce	(coerce (object output-type-spec))
compile	×
compile-file	(compile-file (object))
compile-file-pathname	×
compiled-function-p	×
compiler-macro-function	×
complement	×
complex	×
complexp	×

compute-applicable-methods	×
compute-restarts	×
concatenate	(concatenate (output-type-spec &rest rest))
cond	○
conjugate	×
cons	(cons (object1 object2))
consp	(consp (object))
constantly	×
constantp	×
continue	×
copy-alist	×
copy-list	(copy-list (object))
copy-pprint-dispatch	×
copy-readtable	×
copy-seq	(copy-seq (object))
copy-structure	×
copy-symbol	×
copy-tree	(copy-tree (object))
cos	×
cosh	×
count	(count (a list &key (test (function eql))))
count-if	×
count-if-not	×
ctypecase	×
decf	(decf (a &optional (b 1)))
declaim	○
declare	○
decode-float	×
decode-universal-time	×
defclass	×

defconstant	○
defgeneric	×
define-compiler-macro	×
define-condition	×
define-method-combination	×
define-modify-macro	×
define-setf-expander	×
define-symbol-macro	×
defmacro	○
defmethod	×
defpackage	×
defparameter	○
defsetf	×
defstruct	○
deftype	×
defun	○
defvar	○
delete	(delete (a list &key (test (function eql)) (key (function identity))))
delete-duplicates	(delete-duplicates (list &key (test (function eql)) (key (function identity)) (from-end nil)))
delete-file	×
delete-if	(delete-if (test list &key (key (function identity))))
delete-if-not	(delete-if-not (test list &key (key (function identity))))
delete-package	×
denominator	×
deposit-field	×
describe	×
describe-object	×
destructuring-bind	×
digit-char	×
digit-char-p	(digit-char-p (char &optional (radix 10)))

directory	(directory (object))
directory-namestring	×
disassemble	×
do	○
do*	○
do-all-symbols	×
do-external-symbols	×
do-symbols	×
documentation	×
dolist	○
dotimes	○
double-float	×
dpb	×
dribble	×
ecase	×
echo-stream-input-stream	×
echo-stream-output-stream	×
ed	×
eighth	(eighth (object))
elt	(elt (sequence index))
encode-universal-time	×
endp	×
enough-namestring	(enough-namestring (pathname &optional (default *default-pathname-defaults*)))
ensure-directories-exist	×
ensure-generic-function	×
eq	(eq (object1 object2))
eql	(eql (object1 object2))
equal	(equal (object1 object2))
equalp	(equalp (object1 object2))
error	(error (object &rest arguments))

etypecase	×
eval	(eval (object))
eval-when	○
evenp	×
every	(every (function list))
exp	(exp (object))
export	(export (symbols &optional (package current-package)))
expt	(expt (base-number power-number))
fboundp	×
fceiling	×
fdefinition	×
ffloor	×
fifth	(fifth (object))
file-author	×
file-error-pathname	×
file-length	×
file-namestring	(file-namestring (pathname))
file-position	×
file-string-length	×
file-write-date	(file-write-date (object))
fill	×
fill-pointer	×
find	(find (a b &key (test (function eql)) (key (function identity))))
find-all-symbols	×
find-class	×
find-if	×
find-if-not	×
find-method	×
find-package	(find-package (object))
find-restart	×

find-symbol	×
finish-output	×
first	(first (object))
flet	○
float	(float (object))
float-digits	×
float-precision	×
float-radix	×
float-sign	×
floatp	×
floor	×
fmakunbound	×
force-output	×
format	○
formatter	×
fourth	(fourth (object))
fresh-line	×
fround	×
ftruncate	×
funcall	○
function	○
function-keywords	×
function-lambda-expression	×
functionp	×
gcd	×
gensym	×
gentemp	×
get	(get (symbol indicator &optional default))
get-decoded-time	×
get-dispatch-macro-character	×

get-internal-real-time	×
get-internal-run-time	×
get-macro-character	×
get-output-stream-string	×
get-properties	×
get-setf-expansion	×
get-universal-time	×
getf	×
gethash	(gethash (key hash-table &optional (defaul nil)))
go	×
graphic-char-p	×
handler-bind	×
handler-case	×
hash-table-count	(hash-table-count (object))
hash-table-p	×
hash-table-rehash-size	×
hash-table-rehash-threshold	×
hash-table-size	(hash-table-size (hashtable))
hash-table-test	×
host-namestring	×
identity	(identity (object))
if	○
ignore-errors	×
imagpart	×
import	(import (symbols &optional (package current-package)))
in-package	○
incf	(incf (a &optional (b 1)))
initialize-instance	×
input-stream-p	×
inspect	×

integer-decode-float	×
integer-length	×
integerp	(integerp (object))
interactive-stream-p	×
intern	(intern (symbol &optional (package current-package)))
intersection	(intersection (list1 list2 &key (test (function eql)) (key (function identity))))
invalid-method-error	×
invoke-debugger	×
invoke-restart	×
invoke-restart-interactively	×
isqrt	×
keywordp	(keywordp (object))
labels	○
lambda	○
last	(last (list))
lcm	×
ldb	×
ldb-test	×
ldiff	×
length	(length (object))
let	○
let*	○
lisp-implementation-type	×
lisp-implementation-version	×
list	○
list*	×
list-all-packages	×
list-length	×
listen	×
listp	(listp (object))

load	○
load-logical-pathname-translations	×
load-time-value	×
locally	×
log	(log (object))
logand	(logand (object))
logandc1	×
logandc2	×
logbitp	(logbitp (index integer))
logcount	×
logeqv	×
logical-pathname	×
logical-pathname-translations	×
logior	(logior (&rest integers))
lognand	×
lognor	×
lognot	×
logorc1	×
logorc2	×
logtest	×
logxor	×
long-site-name	×
loop	○
lower-case-p	×
machine-instance	×
machine-type	×
machine-version	×
macro-function	×
macroexpand	×
macroexpand-1	×

macrolet	×
make-array	(make-array (number &key initial-element initial-contents))
make-broadcast-stream	×
make-concatenated-stream	×
make-condition	×
make-dispatch-macro-character	×
make-echo-stream	×
make-hash-table	(make-hash-table (&key (test (function eql)) size))
make-instance	×
make-instances-obsolete	×
make-list	(make-list (size &key initial-element))
make-load-form	×
make-load-form-saving-slots	×
make-package	(make-package (package-name &key use))
make-pathname	×
make-random-state	×
make-sequence	×
make-string	(make-string (number &key initial-element))
make-string-input-stream	×
make-string-output-stream	×
make-symbol	×
make-synonym-stream	×
make-two-way-stream	×
makunbound	×
map	(map (output-type-spec function))
map-into	×
mapc	(mapc (function list))
mapcan	×
mapcar	○
mapcon	×

maphash	(maphash (function list))
mapl	×
maplist	○
mask-field	×
max	○
member	(member (a b &key (test (function eql)) (key (function identity)))
member-if	×
member-if-not	×
merge	×
merge-pathnames	(merge-pathnames (pathname &optional defaults))
method-combination-error	×
method-qualifiers	×
min	○
minusp	(minusp (object))
mismatch	×
mod	(mod (nubmer divisor))
muffle-warning	×
multiple-value-bind	○
multiple-value-call	×
multiple-value-list	×
multiple-value-prog1	×
multiple-value-setq	○
name-char	×
namestring	(namestring (pathname))
nbutlast	×
nconc	(nconc (object1 object2))
nintersection	×
ninth	(ninth (object))
no-applicable-method	×
no-next-method	×

not	(not (object))
notany	×
notevery	×
nreconc	×
nreverse	(nreverse (object))
nset-difference	×
nset-exclusive-or	×
nstring-capitalize	×
nstring-downcase	×
nstring-upcase	×
nsublis	(nsublis (alist sequence &key (test (function eql)) (key (function identity))))
nsubst	(nsubst (newitem olditem sequence &key (test (function eql)) (key (function identity))))
nsubst-if	×
nsubst-if-not	×
nsubstitute	(nsubstitute (newitem olditem sequence &key (test (function eql)) (key (function identity))))
nsubstitute-if	×
nsubstitute-if-not	×
nth	(nth (n list))
nth-value	×
nthcdr	(nthcdr (n list))
null	(null (object))
numberp	(numberp (object))
numerator	×
nunion	×
oddp	(oddp (object))
open	(open (file-name &key direction if-does-not-exist if-exists))
open-stream-p	×
or	○
output-stream-p	×
package-error-package	×

package-name	×
package-nicknames	×
package-shadowing-symbols	×
package-use-list	(package-use-list (object))
package-used-by-list	×
packagep	×
pairlis	×
parse-integer	(parse-integer (a &key (junk-allowed nil)))
parse-namestring	×
pathname	×
pathname-device	×
pathname-directory	(pathname-directory (pathname))
pathname-host	×
pathname-match-p	×
pathname-name	(pathname-name (pathname))
pathname-type	(pathname-type (pathname))
pathname-version	×
pathnamep	(pathnamep (pathname))
peek-char	×
phase	×
plusp	(plusp (object))
pop	(pop (list))
position	(position (item sequence &key (test (function eql)) (key (function identity)) (start 0) end))
position-if	(position-if (test sequence &key (key (function identity)) (start 0) end))
position-if-not	(position-if-not (test sequence &key (key (function identity)) (start 0) end))
pprint	(pprint (object &optional (stream t)))
pprint-dispatch	×
pprint-fill	×
pprint-indent	×
pprint-linear	×

pprint-logical-block	×
pprint-newline	×
pprint-tab	×
pprint-tabular	×
prin1	×
prin1-to-string	×
princ	(princ)
princ-to-string	×
print	(print)
print-not-readable-object	×
print-object	×
print-unreadable-object	×
probe-file	(probe-file (object))
proclaim	×
prog	×
prog*	×
prog1	○
prog2	×
progn	○
progv	×
provide	×
psetf	×
psetq	×
push	(push (object list))
pushnew	(pushnew (object list &key (test (function eql))))
quote	○
random	×
random-state-p	×
rassoc	(rassoc (object a-list &key (test (function eql)) (key (function identity))))
rassoc-if	×

rassoc-if-not	×
rational	×
rationalize	×
rationalp	×
read	(read (stream flg eof-return-value))
read-byte	×
read-char	(read-char (stream flg eof-return-value))
read-char-no-hang	×
read-delimited-list	×
read-from-string	(read-from-string (string &optional flg eof-return-value))
read-line	(read-line (stream eof-error-p eof-value))
read-preserving-whitespace	×
read-sequence	×
readtable-case	×
readtablep	×
realp	×
realpart	×
reduce	(reduce (function sequence))
reinitialize-instance	×
rem	(rem (number divisor))
remf	×
remhash	×
remove	(remove (a list &key (test (function eql)) (key (function identity))))
remove-duplicates	(remove-duplicates (list &key (test (function eql)) (key (function identity)) (from-end nil)))
remove-if	(remove-if (test list &key (key (function identity))))
remove-if-not	(remove-if-not (test list &key (key (function identity))))
remove-method	×
remprop	×
rename-file	×
rename-package	×

replace	×
require	×
rest	(rest (object))
restart-bind	×
restart-case	×
restart-name	×
return	○
return-from	○
revappend	×
reverse	(reverse (list))
room	×
rotatef	×
round	×
row-major-aref	×
rplaca	(rplaca (list element))
rplacd	(rplacd (list element))
sbit	×
scale-float	×
schar	×
search	(search (sequence1 sequence2 &key from-end (test (function eql)) (key (function identity)) (start1 0) (start2 0) end1 end2))
second	(second (object))
set	(set (symbol value))
set-difference	(set-difference (list1 list2 &key (test (function eql)) (key (function identity))))
set-dispatch-macro-character	×
set-exclusive-or	×
set-macro-character	×
set-pprint-dispatch	×
set-syntax-from-char	×
setf	○
setq	○

seventh	(seventh (object))
shadow	×
shadowing-import	×
shared-initialize	×
shiftf	×
short-site-name	×
signal	×
signum	×
simple-bit-vector-p	×
simple-condition-format-arguments	×
simple-condition-format-control	×
simple-string	×
simple-string-p	×
simple-vector-p	×
sin	×
single-float	×
sinh	×
sixth	(sixth (object))
sleep	×
slot-boundp	×
slot-exists-p	×
slot-makunbound	×
slot-missing	×
slot-unbound	×
slot-value	×
software-type	×
software-version	×
some	×
sort	(sort (list predicate &key (key (function identity))))
special-operator-p	×

sqrt	×
stable-sort	(stable-sort (list predicate &key (key (function identity))))
standard-char-p	×
step	×
store-value	×
stream-element-type	×
stream-error-stream	×
stream-external-format	×
streamp	×
string	(string (object))
string-capitalize	(string-capitalize (string &key start end))
string-downcase	(string-downcase (string))
string-equal	(string-equal (string1 string2 &key start1 start2 end1 end2))
string-greaterp	(string-greaterp (string1 string2 &key start1 start2 end1 end2))
string-left-trim	×
string-lessp	(string-lessp (string1 string2 &key start1 start2 end1 end2))
string-not-equal	×
string-not-greaterp	(string-not-greaterp (string1 string2 &key start1 start2 end1 end2))
string-not-lessp	(string-not-lessp (string1 string2 &key start1 start2 end1 end2))
string-right-trim	×
string-trim	(string-trim (char-bag string))
string-upcase	(string-upcase (object))
string/=	×
string<	(string< (string1 string2 &key start1 start2 end1 end2))
string<=	(string<= (string1 string2 &key start1 start2 end1 end2))
string=	(string= (string1 string2 &key start1 start2 end1 end2))
string>	(string> (string1 string2 &key start1 start2 end1 end2))
string>=	(string>= (string1 string2 &key start1 start2 end1 end2))
stringp	(stringp (object))
sublis	(sublis (alist sequence &key (test (function eql)) (key (function identity))))

subseq	(subseq (object star-point &optional (end-point *no-set-arg-value*)))
subsetp	×
subst	(subst (newitem olditem sequence &key (test (function eql)) (key (function identity))))
subst-if	×
subst-if-not	×
substitute	(substitute (newitem olditem sequence &key (test (function eql)) (key (function identity))))
substitute-if	×
substitute-if-not	×
subtypep	(subtypep (type1 type2))
svref	×
sxhash	×
symbol-function	×
symbol-macrolet	×
symbol-name	(symbol-name (symbol))
symbol-package	(symbol-package (object))
symbol-plist	×
symbol-value	(symbol-value (object))
symbolp	(symbolp (object))
synonym-stream-symbol	×
tagbody	×
tailp	×
tan	×
tanh	×
tenth	(tenth (object))
terpri	(terpri (&optional stream))
the	×
third	(third (object))
throw	○
time	○
trace	×

translate-logical-pathname	×
translate-pathname	×
tree-equal	×
truename	×
truncate	(truncate (number &optional divisor))
two-way-stream-input-stream	×
two-way-stream-output-stream	×
type-error-datum	×
type-error-expected-type	×
type-of	(type-of (object))
typecase	○
typep	(typep (object type))
unbound-slot-instance	×
unexport	×
unintern	×
union	(union (list1 list2 &key (test (function eql)) (key (function identity))))
unless	○
unread-char	×
untrace	×
unuse-package	×
unwind-protect	×
update-instance-for-different-class	×
update-instance-for-redefined-class	×
upgraded-array-element-type	×
upgraded-complex-part-type	×
upper-case-p	×
use-package	×
use-value	×
user-homedir-pathname	×
values	○

values-list	×
vector	×
vector-pop	×
vector-push	×
vector-push-extend	×
vectorp	×
warn	×
when	○
wild-pathname-p	×
with-accessors	×
with-compilation-unit	×
with-condition-restarts	×
with-hash-table-iterator	×
with-input-from-string	×
with-open-file	○
with-open-stream	×
with-output-to-string	×
with-package-iterator	×
with-simple-restart	×
with-slots	×
with-standard-io-syntax	×
write	×
write-byte	×
write-char	(write-char (charactor &optional stream))
write-line	×
write-sequence	×
write-string	×
write-to-string	(write-to-string (object))
y-or-n-p	×
yes-or-no-p	×
zerop	(zerop (object))

参考文献

[1] 後藤英一, 井田昌之：“COMMON LISP 言語使用書”, 共立出版 (株),(1986-3)

[2] 西村仁志、隅田：“TR-IT-0289 リスト処理用ライブラリ (LinC) 説明書”, (株) ATR 音声翻訳通信研究所,(1999-2)