

File created: 11-Nov-2023 09:31:38 {DSK}<home>larry>il>medley>lispusers>HELPSYS.;2

edit by: lmm

changes to: (VARS HELPSYSCOMS)
(FNS CLHS.LOOKUP)

previous date: 16-Jun-2023 14:38:33 {DSK}<home>larry>il>medley>lispusers>HELPSYS.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

(RPAQQ HELPSYSCOMS

```
((FILES DINFO HASH)
 (RECORDS IRMREFERENCE)
 (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (LOADCOMP)
                                         DINFO HASH))

 [COMS (COMMANDS "man")
       (FNS HELPSYS IRM.LOOKUP GENERIC.MAN.LOOKUP IRM.SMART.LOOKUP IRM.RESET)
       (INITVARS (IRM.HOST&DIR
                 (IRM.HASHFILE.NAME))
                (GLOBALVARS IRM.HOST&DIR IRM.HASHFILE.NAME)
                (DECLARE%: DONTEVAL@LOAD DOCOPY (P (MOVD 'HELPSYS 'XHELPSYS NIL T))
```

::: Common Lisp HyperSpec lookup

```
(COMS (FNS CLHS.INDEX CLHS.LOOKUP CLHS.OPENER REPO.LOOKUP)
      (VARS CLHS.INDEX)
      [INITVARS (CLHS.ROOT.URL "https://interlisp.org/clhs/")
                (CLHS.INDEX)
                (CLHS.OPENER)
                (HELPSYS.REPO.TYPES ' (FNS FUNCTIONS VARS VARIABLES)]
      (GLOBALVARS CLHS.INDEX CLHS.OPENER HELPSYS.REPO.TYPES CLHS.ROOT.URL))
 (COMS
```

::: Interface to DInfo

```
(FNS IRM.GET.DINFOGRAPH IRM.DISPLAY.REF)
(FUNCTIONS IRM.LOAD-GRAPH)
(ADDVARS (DINFO.GRAPHS ("Interlisp-D Reference Manual" (IRM.GET.DINFOGRAPH T)
 (INITVARS (IRM.DINFOGRAPH))
 (GLOBALVARS IRM.DINFOGRAPH)
 (DECLARE%: DONTEVAL@LOAD DOCOPY (P (COND (IRM.HOST&DIR (SETQ IRM.DINFOGRAPH (IRM.LOAD-GRAPH)
```

::: Cross reference imageobj

```
(FNS IRM.DISPLAY.CREF IRM.CREF.BOX IRM.PUT.CREF IRM.GET.CREF IRM.CREF.BUTTONEVENTFN)
[INITVARS (IRM.CREF.FONT (FONTCREATE 'MODERN 8 'MRR))
          (\IRM.CREF.IMAGEFNS (IMAGEFNSCREATE (FUNCTION IRM.DISPLAY.CREF)
                                             (FUNCTION IRM.CREF.BOX)
                                             (FUNCTION IRM.PUT.CREF)
                                             (FUNCTION IRM.GET.CREF)
                                             (FUNCTION NIL)
                                             (FUNCTION IRM.CREF.BUTTONEVENTFN))
          (GLOBALVARS IRM.CREF.FONT \IRM.CREF.IMAGEFNS))
 (COMS
```

::: Internal functions and variables

```
(FNS \IRM.GET.REF \IRM.SMART.REF \IRM.CHOOSE.REF \IRM.WILD.REF \IRM.WILDCARD \IRM.WILD.MATCH
     \IRM.GET.HASHFILE \IRM.GET.KEYWORDS)
(INITVARS (\IRM.HASHFILE)
          (\IRM.KEYWORDS))
(GLOBALVARS \IRM.HASHFILE \IRM.KEYWORDS)
(FUNCTIONS \IRM.AROUND-EXIT)
(ADDVARS (AROUNDEXITFNS \IRM.AROUND-EXIT))
(PROP (FILETYPE)
      HELPSYS)))
```

(FILESLOAD DINFO HASH)

(DECLARE%: EVAL@COMPILE

(RECORD IRMREFERENCE

;; A reference to something in the IRM. There is a list of these for each entry in the index of the IRM. Each element of the list corresponds to
;; one of the page references. These lists are stored under the ITEM in a hash file.

(TYPE	; The type of index entry -- typically a capitalized symbol in IL,
	; eg. il: Functions . Yes, it's ugly.
ITEM	; The name indexed
PRIMARYFLG	; True iff this is the primary reference for this name/type
NODE	; The ID of the node in the IRM DInfo graph containing this
	; reference

```

CH# ; The character number of the beginning of the reference. If
; unspecified we search for the first existence of NAME in the
; text of the node.

)
(SYSTEM)
)

```

```

(DECLARE%: EVAL@COMPILE DONTCOPY

(FILELOAD (LOADCOMP)
DINFO HASH)

)

```

```

(DEFCOMMAND "man" (ENTRY) "Lookup ENTRY in the IRM."
(GENERIC.MAN.LOOKUP ENTRY))

```

(DEFINEQ

(HELPSYS

[LAMBDA (FN PROPS)

```

; Edited 24-Aug-2022 16:17 by larry
; Edited 13-Aug-2022 22:35 by Imm
(* drc%: "20-Jan-86 18:05")

```

```

(if (NOT IRM.HOST&DIR)
then (PROMPTPRINT "HELPSYS is unavailable. Set IRM.HOST&DIR.")
NIL
else (SELECTQ PROPS
(ARGS

```

(* HELPSYS is called by SMARTARGLIST to get args, but this implementation does not support that.)

```

NIL)
(FromDEdit (IRM.LOOKUP (if (LISTP FN)
then (CAR FN)
else FN))
NIL)

```

(* from ? under EditCom)

```

NIL)
(* called by TTYIN <actually XHELPSYS is...> when FN...?
<CR> is typed.)

```

```

(if (FGETD FN)
then (GENERIC.MAN.LOOKUP FN NIL 'Function)
elseif (for MACRO.TYPE in MACROPROPS thereis (GETPROP FN MACRO.TYPE))
then (IRM.LOOKUP FN 'Macro IRMWINDOW)
elseif (SELECTQ (CAR (GETPROP FN 'CLISPCWORD))
(NIL)
(FORWARD (IRM.LOOKUP FN 'I.S.Operator))
(RECORDTRAN (IRM.LOOKUP FN 'RecordOperator))
(PROGN (IRM.LOOKUP FN NIL)))
else (BEEP)))
NIL])

```

(IRM.LOOKUP

[LAMBDA (KEYWORD TYPE GRAPH SMARTFLG)

```

; Edited 24-Aug-2022 16:32 by larry
; Edited 19-Aug-2022 19:43 by Imm
(* drc%: "17-Jan-86 14:09")

```

```

;;; Does a lookup in the IRM index for KEYWORD (optionally of TYPE) and visits the DInfo node in GRAPH containing the reference. If SMARTFLG is
;;; non-NIL, wildcards will be enabled. GRAPH defaults to IRM.DINFOGRAPH.

```

```

(PROG [(REF (if SMARTFLG
then (IRM.SMART.REF KEYWORD)
else (IRM.GET.REF KEYWORD TYPE]
(if (NULL REF)
then (RETURN))
(LET* [(GRAPH (if (type? DINFOGRAPH GRAPH)
then GRAPH
else (IRM.GET.DINFOGRAPH))]
(KEYWORD (MKATOM (U-CASE KEYWORD)))
(TYPE (MKATOM TYPE))
(WINDOW (fetch (DINFOGRAPH WINDOW) of GRAPH))
(MONITORLOCK (DINFOGRAPHPROP GRAPH 'MONITORLOCK]
(OPENW WINDOW)
(if (OBTAIN.MONITORLOCK MONITORLOCK T)
then (RESETLST
(RESETSAVE (TTYDISPLAYSTREAM (GETPROMPTWINDOW WINDOW)))
(RESETSAVE NIL (LIST 'RELEASE.MONITORLOCK MONITORLOCK))
(IRM.DISPLAY.REF REF GRAPH)
(LIST REF))
else (FLASHWINDOW WINDOW)
NIL])

```

(GENERIC.MAN.LOOKUP

[LAMBDA (KEYWORD GRAPH TYPE)

```

; Edited 27-Aug-2022 12:15 by larry
; Edited 24-Aug-2022 22:35 by larry

```



```

"]
      (SETQ TERM (SUBSTRING LINE POSTERM (CL:1- POSENDTERM)
      collect (CONS (for SUBST in '("&" "&")) when (EQ 1 (STRPOS (CAR SUBST)
                                                              TERM))
                do [SETQ TERM (CONCAT (CADR SUBST)
                                        (SUBSTRING TERM (PLUS 1 (NCHARS (CAR SUBST)
                                                              finally (RETURN TERM))
      (if (SETQ POSFRAG (STRPOS "#" LINE POSLINK POSENDLINK))
          then (LIST (SUBSTRING LINE POSLINK (CL:1- POSFRAG))
                    (SUBSTRING LINE (CL:1+ POSFRAG)
                                     (CL:1- POSENDLINK)))
          else (LIST (SUBSTRING LINE POSLINK (CL:1- POSENDLINK))

```

(CLHS.LOOKUP

[LAMBDA (ENTRY PHASES)

; Edited 11-Nov-2023 09:11 by Imm
 ; Edited 12-Oct-2022 18:32 by FGH
 ; Edited 24-Aug-2022 17:08 by larry

```

(LET [(OPENER (ShellBrowser))
      (URL NIL)
      (POS
       (ENTRY (L-CASE (MKSTRING ENTRY)
                (for PHASE in (OR PHASES '(1 2 3))
                do ;; three phases: exact match, initial match, partial match
                  (for X in (CLHS.INDEX) when (SELECTQ PHASE
                                                    (1 (STREQVAL ENTRY (CAR X)))
                                                    (2 [AND (EQ (STRPOS ENTRY (CAR X))
                                                                1)
                                                                (NOT (STREQVAL ENTRY (CAR X))
                                                                 (3 (AND (SETQ POS (STRPOS ENTRY (CAR X))
                                                                (NEQ POS 1)))
                                                    NIL)
                  join (SETQ URL (CONCAT CLHS.ROOT.URL "Body/" (CADR X)
                                         (if (CADDR X)
                                             then (CONCAT "#" (CADDR X))
                                             else "")))
                (if (EQUAL OPENER "lynx")
                    then ;; Need to quote URL because shell eats #
                      (CHAT 'SHELL NIL (CONCAT OPENER " " URL " "
                                                ")))
                    else (ShellCommand (CONCAT OPENER " " URL " " " > /tmp/clhs-warnings-$$$.txt 2>&1")
                                         T))
                (RETURN))
      (AND URL (RETURN (LIST URL])

```

(CLHS.OPENER

[LAMBDA NIL

; Edited 20-Aug-2022 09:38 by larry
 ; Edited 20-Aug-2022 09:20 by Imm
 ; Edited 16-Aug-2022 16:50 by Imm
 ; Edited 16-Aug-2022 12:22 by larry
 ; Edited 15-Aug-2022 09:14 by Imm

```

(OR CLHS.OPENER (if (INFILEP "{UNIX}/usr/bin/wslview")
                   then ;; windows with WSL
                     "wslview"
                   elseif (STRPOS "darwin" (OR (UNIX-GETENV "OSTYPE")
                                                (UNIX-GETENV "PATH")))
                     then ;; MacOS
                       "open"
                   elseif (INFILEP "{UNIX}/usr/bin/lynx")
                     then (if (INFILEP "{UNIX}/usr/bin/xterm")
                              then "xterm -e lynx"
                              else "lynx")
                   else "git web--browse]))

```

(REPO.LOOKUP

[LAMBDA (ENTRY TYPES)

; Edited 13-Jan-2023 10:46 by Imm
 ; Edited 16-Aug-2022 16:26 by Imm

```

(for FL in (UNION (WHEREIS ENTRY (OR TYPES HELPSYS.REPO.TYPES)
                    T)
              (LIST ENTRY))
  bind POS FND when [SETQ FND (OR (FINDFILE-WITH-EXTENSIONS FL NIL '(TEDIT TXT TED))
                                (AND (SETQ POS (STRPOS "-" FL))
                                     (FINDFILE-WITH-EXTENSIONS (SUBSTRING FL 1 (CL:1- POS))
                                                                NIL
                                                                '(TEDIT TXT TTY TED))
                                :DIRECTION :INPUT)
  join (CL:WITH-OPEN-FILE (STR (PATHNAME FND)
                              (CL:WHEN (SETQ POS (FFILEPOS ENTRY STR))
                                        (TEDIT-SEE STR NIL NIL (CL:FORMAT NIL "~a [~a]" FL ENTRY))
                                        (LIST FL)))]

```

)

(RPAQQ CLHS.INDEX

```

(("&allow-other-keys" "03_da.htm" "AMallow-other-keys")
("&aux" "03_da.htm" "AMaux")
("&body" "03_dd.htm" "AMbody")
("&environment" "03_dd.htm" "AMenvironment")
("&key" "03_da.htm" "AMkey")
("&optional" "03_da.htm" "AMoptional")
("&rest" "03_da.htm" "AMrest")
("&whole" "03_dd.htm" "AMwhole")
("**" "a_st.htm" "ST")
(***) "v__stst_.htm" "STST")
(***) "v__stst_.htm" "STSTST")
(*break-on-signals* "v_break_.htm" "STbreak-on-signalsST")
(*compile-file-pathname* "v_cmp_fi.htm" "STcompile-file-pathnameST")
(*compile-file-truename* "v_cmp_fi.htm" "STcompile-file-truenameST")
(*compile-print* "v_cmp_pr.htm" "STcompile-printST")
(*compile-verbose* "v_cmp_pr.htm" "STcompile-verboseST")
(*debug-io* "v_debug_.htm" "STdebug-ioST")
(*debugger-hook* "v_debugg.htm" "STdebugger-hookST")
(*default-pathname-defaults* "v_defaul.htm" "STdefault-pathname-defaultsST")
(*error-output* "v_debug_.htm" "STerror-outputST")
(*features* "v_featur.htm" "STfeaturesST")
(*gensym-counter* "v_gensym.htm" "STgensym-counterST")
(*load-pathname* "v_ld_pns.htm" "STload-pathnameST")
(*load-print* "v_ld_prs.htm" "STload-printST")
(*load-truename* "v_ld_pns.htm" "STload-truenameST")
(*load-verbose* "v_ld_prs.htm" "STload-verboseST")
(*macroexpand-hook* "v_mexp_h.htm" "STmacroexpand-hookST")
(*modules* "v_module.htm" "STmodulesST")
(*package* "v_pkg.htm" "STpackageST")
(*print-array* "v_pr_ar.htm" "STprint-arrayST")
(*print-base* "v_pr_bas.htm" "STprint-baseST")
(*print-case* "v_pr_cas.htm" "STprint-caseST")
(*print-circle* "v_pr_cir.htm" "STprint-circleST")
(*print-escape* "v_pr_esc.htm" "STprint-escapeST")
(*print-gensym* "v_pr_gen.htm" "STprint-gensymST")
(*print-length* "v_pr_lev.htm" "STprint-lengthST")
(*print-level* "v_pr_lev.htm" "STprint-levelST")
(*print-lines* "v_pr_lin.htm" "STprint-linesST")
(*print-miser-width* "v_pr_mis.htm" "STprint-miser-widthST")
(*print-pprint-dispatch* "v_pr_ppr.htm" "STprint-pprint-dispatchST")
(*print-pretty* "v_pr_pre.htm" "STprint-prettyST")
(*print-radix* "v_pr_bas.htm" "STprint-radixST")
(*print-readably* "v_pr_rda.htm" "STprint-readablyST")
(*print-right-margin* "v_pr_rig.htm" "STprint-right-marginST")
(*query-io* "v_debug_.htm" "STquery-ioST")
(*random-state* "v_rnd_st.htm" "STrandom-stateST")
(*read-base* "v_rd_bas.htm" "STread-baseST")
(*read-default-float-format* "v_rd_def.htm" "STread-default-float-formatST")
(*read-eval* "v_rd_eva.htm" "STread-evalST")
(*read-suppress* "v_rd_sup.htm" "STread-suppressST")
(*readtable* "v_rdtabl.htm" "STreadtableST")
(*standard-input* "v_debug_.htm" "STstandard-inputST")
(*standard-output* "v_debug_.htm" "STstandard-outputST")
(*terminal-io* "v_termin.htm" "STterminal-ioST")
(*trace-output* "v_debug_.htm" "STtrace-outputST")
(+" "a_pl.htm" "PL")
(++) "v_pl_plp.htm" "PLPL")
(+++) "v_pl_plp.htm" "PLPLPL")
(-" "a_.htm" "-")
(/" "a_sl.htm" "SL")
(// "v_sl_sls.htm" "SLSL")
(/// "v_sl_sls.htm" "SLSLSL")
(/=" "f_eq_sle.htm" "SLEQ")
(1+ "f_1pl_1_.htm" "1PL")
(1- "f_1pl_1_.htm" "1-")
(&lt;" "f_eq_sle.htm" "LT")
(&lt;=" "f_eq_sle.htm" "LTEQ")
(=" "f_eq_sle.htm" "EQ")
(&gt;" "f_eq_sle.htm" "GT")
(&gt;=" "f_eq_sle.htm" "GTEQ")
("abort" "a_abort.htm" "abort")
("abs" "f_abs.htm" "abs")
("acons" "f_acons.htm" "acons")
("acos" "f_asin_.htm" "acos")
("acosh" "f_sinh_.htm" "acosh")
("add-method" "f_add_me.htm" "add-method")
("adjoin" "f_adjoin.htm" "adjoin")
("adjust-array" "f_adjust.htm" "adjust-array")
("adjustable-array-p" "f_adju_1.htm" "adjustable-array-p")
("allocate-instance" "f_alloc_.htm" "allocate-instance")
("alpha-char-p" "f_alpha_.htm" "alpha-char-p")
("alphanumericp" "f_alphan.htm" "alphanumericp")
("and" "a_and.htm" "and")
("append" "f_append.htm" "append")

```

```

("apply" "f_apply.htm" "apply")
("apropos" "f_apropo.htm" "apropos")
("apropos-list" "f_apropo.htm" "apropos-list")
("aref" "f_aref.htm" "aref")
("arithmetic-error" "e_arithm.htm" "arithmetic-error")
("arithmetic-error-operands" "f_arithm.htm" "arithmetic-error-operands")
("arithmetic-error-operation" "f_arithm.htm" "arithmetic-error-operation")
("array" "t_array.htm" "array")
("array-dimension" "f_ar_dim.htm" "array-dimension")
("array-dimension-limit" "v_ar_dim.htm" "array-dimension-limit")
("array-dimensions" "f_ar_d_l.htm" "array-dimensions")
("array-displacement" "f_ar_dis.htm" "array-displacement")
("array-element-type" "f_ar_ele.htm" "array-element-type")
("array-has-fill-pointer-p" "f_ar_has.htm" "array-has-fill-pointer-p")
("array-in-bounds-p" "f_ar_in_.htm" "array-in-bounds-p")
("array-rank" "f_ar_ran.htm" "array-rank")
("array-rank-limit" "v_ar_ran.htm" "array-rank-limit")
("array-row-major-index" "f_ar_row.htm" "array-row-major-index")
("array-total-size" "f_ar_tot.htm" "array-total-size")
("array-total-size-limit" "v_ar_tot.htm" "array-total-size-limit")
("arrayp" "f_arrayp.htm" "arrayp")
("ash" "f_ash.htm" "ash")
("asin" "f_asin_.htm" "asin")
("asinh" "f_sinh_.htm" "asinh")
("assert" "m_assert.htm" "assert")
("assoc" "f_assocc.htm" "assoc")
("assoc-if" "f_assocc.htm" "assoc-if")
("assoc-if-not" "f_assocc.htm" "assoc-if-not")
("atan" "f_asin_.htm" "atan")
("atanh" "f_sinh_.htm" "atanh")
("atom" "a_atom.htm" "atom")
("base-char" "t_base_c.htm" "base-char")
("base-string" "t_base_s.htm" "base-string")
("bignum" "t_bignum.htm" "bignum")
("bit" "a_bit.htm" "bit")
("bit-and" "f_bt_and.htm" "bit-and")
("bit-andc1" "f_bt_and.htm" "bit-andc1")
("bit-andc2" "f_bt_and.htm" "bit-andc2")
("bit-eqv" "f_bt_and.htm" "bit-eqv")
("bit-ior" "f_bt_and.htm" "bit-ior")
("bit-nand" "f_bt_and.htm" "bit-nand")
("bit-nor" "f_bt_and.htm" "bit-nor")
("bit-not" "f_bt_and.htm" "bit-not")
("bit-orc1" "f_bt_and.htm" "bit-orc1")
("bit-orc2" "f_bt_and.htm" "bit-orc2")
("bit-vector" "t_bt_vec.htm" "bit-vector")
("bit-vector-p" "f_bt_vec.htm" "bit-vector-p")
("bit-xor" "f_bt_and.htm" "bit-xor")
("block" "s_block.htm" "block")
("boole" "f_boole.htm" "boole")
("boole-1" "v_b_1_b.htm" "boole-1")
("boole-2" "v_b_1_b.htm" "boole-2")
("boole-and" "v_b_1_b.htm" "boole-and")
("boole-andc1" "v_b_1_b.htm" "boole-andc1")
("boole-andc2" "v_b_1_b.htm" "boole-andc2")
("boole-c1" "v_b_1_b.htm" "boole-c1")
("boole-c2" "v_b_1_b.htm" "boole-c2")
("boole-clr" "v_b_1_b.htm" "boole-clr")
("boole-eqv" "v_b_1_b.htm" "boole-eqv")
("boole-ior" "v_b_1_b.htm" "boole-ior")
("boole-nand" "v_b_1_b.htm" "boole-nand")
("boole-nor" "v_b_1_b.htm" "boole-nor")
("boole-orc1" "v_b_1_b.htm" "boole-orc1")
("boole-orc2" "v_b_1_b.htm" "boole-orc2")
("boole-set" "v_b_1_b.htm" "boole-set")
("boole-xor" "v_b_1_b.htm" "boole-xor")
("boolean" "t_ban.htm" "boolean")
("both-case-p" "f_upper_.htm" "both-case-p")
("boundp" "f_boundp.htm" "boundp")
("break" "f_break.htm" "break")
("broadcast-stream" "t_broadc.htm" "broadcast-stream")
("broadcast-stream-streams" "f_broadc.htm" "broadcast-stream-streams")
("built-in-class" "t_built_.htm" "built-in-class")
("butlast" "f_butlas.htm" "butlast")
("byte" "f_by_by.htm" "byte")
("byte-position" "f_by_by.htm" "byte-position")
("byte-size" "f_by_by.htm" "byte-size")
("caaaar" "f_car_c.htm" "caaaar")
("caaaadr" "f_car_c.htm" "caaaadr")
("caaar" "f_car_c.htm" "caaar")
("caadar" "f_car_c.htm" "caadar")
("caaddr" "f_car_c.htm" "caaddr")
("caadr" "f_car_c.htm" "caadr")
("caar" "f_car_c.htm" "caar")
("cadaar" "f_car_c.htm" "cadaar")
("cadadr" "f_car_c.htm" "cadadr")
("cadar" "f_car_c.htm" "cadar")

```

```

("caddar" "f_car_c.htm" "caddar")
("caddr" "f_car_c.htm" "caddr")
("cadr" "f_car_c.htm" "cadr")
("call-arguments-limit" "v_call_a.htm" "call-arguments-limit")
("call-method" "m_call_m.htm" "call-method")
("call-next-method" "f_call_n.htm" "call-next-method")
("car" "f_car_c.htm" "car")
("case" "m_case_.htm" "case")
("catch" "s_catch.htm" "catch")
("ccase" "m_case_.htm" "ccase")
("cdaaar" "f_car_c.htm" "cdaaar")
("cdaadr" "f_car_c.htm" "cdaadr")
("cdaar" "f_car_c.htm" "cdaar")
("cdadar" "f_car_c.htm" "cdadar")
("cdaddr" "f_car_c.htm" "cdaddr")
("cdadr" "f_car_c.htm" "cdadr")
("cdar" "f_car_c.htm" "cdar")
("cddaar" "f_car_c.htm" "cddaar")
("cddadr" "f_car_c.htm" "cddadr")
("cddar" "f_car_c.htm" "cddar")
("cddddar" "f_car_c.htm" "cddddar")
("cddddr" "f_car_c.htm" "cddddr")
("cdddr" "f_car_c.htm" "cdddr")
("cddr" "f_car_c.htm" "cddr")
("cdr" "f_car_c.htm" "cdr")
("ceiling" "f_floorc.htm" "ceiling")
("cell-error" "e_cell_e.htm" "cell-error")
("cell-error-name" "f_cell_e.htm" "cell-error-name")
("cerror" "f_cerror.htm" "cerror")
("change-class" "f_chg_cl.htm" "change-class")
("char" "f_char_.htm" "char")
("char-code" "f_char_c.htm" "char-code")
("char-code-limit" "v_char_c.htm" "char-code-limit")
("char-downcase" "f_char_u.htm" "char-downcase")
("char-equal" "f_chareq.htm" "char-equal")
("char-greaterp" "f_chareq.htm" "char-greaterp")
("char-int" "f_char_i.htm" "char-int")
("char-lessp" "f_chareq.htm" "char-lessp")
("char-name" "f_char_n.htm" "char-name")
("char-not-equal" "f_chareq.htm" "char-not-equal")
("char-not-greaterp" "f_chareq.htm" "char-not-greaterp")
("char-not-lessp" "f_chareq.htm" "char-not-lessp")
("char-upcase" "f_char_u.htm" "char-upcase")
("char/= " "f_chareq.htm" "charSLEQ")
("char< " "f_chareq.htm" "charLT")
("char<=" "f_chareq.htm" "charLTEQ")
("char=" "f_chareq.htm" "charEQ")
("char> " "f_chareq.htm" "charGT")
("char>=" "f_chareq.htm" "charGTEQ")
("character" "a_ch.htm" "character")
("characterp" "f_chp.htm" "characterp")
("check-type" "m_check_.htm" "check-type")
("cis" "f_cis.htm" "cis")
("class" "t_class.htm" "class")
("class-name" "f_class_.htm" "class-name")
("class-of" "f_clas_l.htm" "class-of")
("clear-input" "f_clear_.htm" "clear-input")
("clear-output" "f_finish.htm" "clear-output")
("close" "f_close.htm" "close")
("clrhash" "f_clrhas.htm" "clrhash")
("code-char" "f_code_c.htm" "code-char")
("coerce" "f_coerce.htm" "coerce")
("compilation-speed" "d_optimi.htm" "compilation-speed")
("compile" "f_cmp.htm" "compile")
("compile-file" "f_cmp_fi.htm" "compile-file")
("compile-file-pathname" "f_cmp_l.htm" "compile-file-pathname")
("compiled-function" "t_cmpd_f.htm" "compiled-function")
("compiled-function-p" "f_cmpd_f.htm" "compiled-function-p")
("compiler-macro" "f_docume.htm" "compiler-macro")
("compiler-macro-function" "f_cmp_ma.htm" "compiler-macro-function")
("complement" "f_comple.htm" "complement")
("complex" "a_comple.htm" "complex")
("complexp" "f_comp_3.htm" "complexp")
("compute-applicable-methods" "f_comput.htm" "compute-applicable-methods")
("compute-restarts" "f_comp_1.htm" "compute-restarts")
("concatenate" "f_concat.htm" "concatenate")
("concatenated-stream" "t_concat.htm" "concatenated-stream")
("concatenated-stream-streams" "f_conc_1.htm" "concatenated-stream-streams")
("cond" "m_cond.htm" "cond")
("condition" "e_cnd.htm" "condition")
("conjugate" "f_conjug.htm" "conjugate")
("cons" "a_cons.htm" "cons")
("consp" "f_consp.htm" "consp")
("constantly" "f_cons_1.htm" "constantly")
("constantp" "f_consta.htm" "constantp")
("continue" "a_contin.htm" "continue")

```

```

("control-error" "e_contro.htm" "control-error")
("copy-alist" "f_cp_ali.htm" "copy-alist")
("copy-list" "f_cp_lis.htm" "copy-list")
("copy-pprint-dispatch" "f_cp_ppr.htm" "copy-pprint-dispatch")
("copy-readtable" "f_cp_rdt.htm" "copy-readtable")
("copy-seq" "f_cp_seq.htm" "copy-seq")
("copy-structure" "f_cp_stu.htm" "copy-structure")
("copy-symbol" "f_cp_sym.htm" "copy-symbol")
("copy-tree" "f_cp_tre.htm" "copy-tree")
("cos" "f_sin_c.htm" "cos")
("cosh" "f_sinh_.htm" "cosh")
("count" "f_countc.htm" "count")
("count-if" "f_countc.htm" "count-if")
("count-if-not" "f_countc.htm" "count-if-not")
("ctypecase" "m_tpcase.htm" "ctypecase")
("debug" "d_optimi.htm" "debug")
("defc" "m_incf_.htm" "defc")
("declaim" "m_declai.htm" "declaim")
("declaration" "d_declar.htm" "declaration")
("declare" "s_declar.htm" "declare")
("decode-float" "f_dec_fl.htm" "decode-float")
("decode-universal-time" "f_dec_un.htm" "decode-universal-time")
("defclass" "m_defcla.htm" "defclass")
("defconstant" "m_defcon.htm" "defconstant")
("defgeneric" "m_defgen.htm" "defgeneric")
("define-compiler-macro" "m_define.htm" "define-compiler-macro")
("define-condition" "m_defi_5.htm" "define-condition")
("define-method-combination" "m_defi_4.htm" "define-method-combination")
("define-modify-macro" "m_defi_2.htm" "define-modify-macro")
("define-setf-expander" "m_defi_3.htm" "define-setf-expander")
("define-symbol-macro" "m_defi_1.htm" "define-symbol-macro")
("defmacro" "m_defmac.htm" "defmacro")
("defmethod" "m_defmet.htm" "defmethod")
("defpackage" "m_defpkg.htm" "defpackage")
("defparameter" "m_defpar.htm" "defparameter")
("defsetf" "m_defset.htm" "defsetf")
("defstruct" "m_defstr.htm" "defstruct")
("deftype" "m_deftp.htm" "deftype")
("defun" "m_defun.htm" "defun")
("defvar" "m_defpar.htm" "defvar")
("delete" "f_rm_rm.htm" "delete")
("delete-duplicates" "f_rm_dup.htm" "delete-duplicates")
("delete-file" "f_del_fi.htm" "delete-file")
("delete-if" "f_rm_rm.htm" "delete-if")
("delete-if-not" "f_rm_rm.htm" "delete-if-not")
("delete-package" "f_del_pk.htm" "delete-package")
("denominator" "f_numera.htm" "denominator")
("deposit-field" "f_deposi.htm" "deposit-field")
("describe" "f_descri.htm" "describe")
("describe-object" "f_desc_1.htm" "describe-object")
("destructuring-bind" "m_destru.htm" "destructuring-bind")
("digit-char" "f_digit_.htm" "digit-char")
("digit-char-p" "f_digi_1.htm" "digit-char-p")
("directory" "f_dir.htm" "directory")
("directory-namestring" "f_namest.htm" "directory-namestring")
("disassemble" "f_disass.htm" "disassemble")
("division-by-zero" "e_divisi.htm" "division-by-zero")
("do" "m_do_do.htm" "do")
("do*" "m_do_do.htm" "do*")
("do-all-symbols" "m_do_sym.htm" "do-all-symbols")
("do-external-symbols" "m_do_sym.htm" "do-external-symbols")
("do-symbols" "m_do_sym.htm" "do-symbols")
("documentation" "f_docume.htm" "documentation")
("dolist" "m_dolist.htm" "dolist")
("dotimes" "m_dotime.htm" "dotimes")
("double-float" "t_short_.htm" "double-float")
("double-float-epsilon" "v_short_.htm" "double-float-epsilon")
("double-float-negative-epsilon" "v_short_.htm" "double-float-negative-epsilon")
("dpb" "f_dpb.htm" "dpb")
("dribble" "f_dribbl.htm" "dribble")
("dynamic-extent" "d_dynami.htm" "dynamic-extent")
("ecase" "m_case_.htm" "ecase")
("echo-stream" "t_echo_s.htm" "echo-stream")
("echo-stream-input-stream" "f_echo_s.htm" "echo-stream-input-stream")
("echo-stream-output-stream" "f_echo_s.htm" "echo-stream-output-stream")
("ed" "f_ed.htm" "ed")
("eighth" "f_firstc.htm" "eighth")
("elt" "f_elt.htm" "elt")
("encode-universal-time" "f_encode.htm" "encode-universal-time")
("end-of-file" "e_end_of.htm" "end-of-file")
("endp" "f_endp.htm" "endp")
("enough-namestring" "f_namest.htm" "enough-namestring")
("ensure-directories-exist" "f_ensu_1.htm" "ensure-directories-exist")
("ensure-generic-function" "f_ensure.htm" "ensure-generic-function")
("eq" "f_eq.htm" "eq")
("eql" "a_eql.htm" "eql")
("equal" "f_equal.htm" "equal")

```



```

(equalp "f_equalp.htm" "equalp")
(error "a_error.htm" "error")
(etypecase "m_tpcase.htm" "etypecase")
(eval "f_eval.htm" "eval")
(eval-when "s_eval_w.htm" "eval-when")
(evenp "f_evenpc.htm" "evenp")
(every "f_everyc.htm" "every")
(exp "f_exp_e.htm" "exp")
(export "f_export.htm" "export")
(expt "f_exp_e.htm" "expt")
(extended-char "t_extend.htm" "extended-char")
(fboundp "f_fbound.htm" "fboundp")
(fceiling "f_floorc.htm" "fceiling")
(fdefinition "f_fdefin.htm" "fdefinition")
(ffloor "f_floorc.htm" "ffloor")
(fifth "f_firstc.htm" "fifth")
(file-author "f_file_a.htm" "file-author")
(file-error "e_file_e.htm" "file-error")
(file-error-pathname "f_file_e.htm" "file-error-pathname")
(file-length "f_file_l.htm" "file-length")
(file-namestring "f_namest.htm" "file-namestring")
(file-position "f_file_p.htm" "file-position")
(file-stream "t_file_s.htm" "file-stream")
(file-string-length "f_file_s.htm" "file-string-length")
(file-write-date "f_file_w.htm" "file-write-date")
(fill "f_fill.htm" "fill")
(fill-pointer "f_fill_p.htm" "fill-pointer")
(find "f_find.htm" "find")
(find-all-symbols "f_find_a.htm" "find-all-symbols")
(find-class "f_find_c.htm" "find-class")
(find-if "f_find.htm" "find-if")
(find-if-not "f_find.htm" "find-if-not")
(find-method "f_find_m.htm" "find-method")
(find-package "f_find_p.htm" "find-package")
(find-restart "f_find_r.htm" "find-restart")
(find-symbol "f_find_s.htm" "find-symbol")
(finish-output "f_finish.htm" "finish-output")
(first "f_firstc.htm" "first")
(fixnum "t_fixnum.htm" "fixnum")
(flet "s_flet_.htm" "flet")
(float "a_float.htm" "float")
(float-digits "f_dec_fl.htm" "float-digits")
(float-precision "f_dec_fl.htm" "float-precision")
(float-radix "f_dec_fl.htm" "float-radix")
(float-sign "f_dec_fl.htm" "float-sign")
(floating-point-inexact "e_floa_1.htm" "floating-point-inexact")
(floating-point-invalid-operation "e_floa1i.htm" "floating-point-invalid-operation")
(floating-point-overflow "e_floa_2.htm" "floating-point-overflow")
(floating-point-underflow "e_floa_3.htm" "floating-point-underflow")
(floatp "f_floatp.htm" "floatp")
(floor "f_floorc.htm" "floor")
(fmakunbound "f_fmakun.htm" "fmakunbound")
(force-output "f_finish.htm" "force-output")
(format "f_format.htm" "format")
(formatter "m_format.htm" "formatter")
(fourth "f_firstc.htm" "fourth")
(fresh-line "f_terpri.htm" "fresh-line")
(fround "f_floorc.htm" "fround")
(ftruncate "f_floorc.htm" "ftruncate")
(ftype "d_ftype.htm" "ftype")
(funcall "f_funcal.htm" "funcall")
(function "a_fn.htm" "function")
(function-keywords "f_fn_kwd.htm" "function-keywords")
(function-lambda-expression "f_fn_lam.htm" "function-lambda-expression")
(functionp "f_fnp.htm" "functionp")
(gcd "f_gcd.htm" "gcd")
(generic-function "t_generi.htm" "generic-function")
(gensym "f_gensym.htm" "gensym")
(gentemp "f_gentem.htm" "gentemp")
(get "f_get.htm" "get")
(get-decoded-time "f_get_un.htm" "get-decoded-time")
(get-dispatch-macro-character "f_set_1.htm" "get-dispatch-macro-character")
(get-internal-real-time "f_get_in.htm" "get-internal-real-time")
(get-internal-run-time "f_get_1.htm" "get-internal-run-time")
(get-macro-character "f_set_ma.htm" "get-macro-character")
(get-output-stream-string "f_get_ou.htm" "get-output-stream-string")
(get-properties "f_get_pr.htm" "get-properties")
(get-setf-expansion "f_get_se.htm" "get-setf-expansion")
(get-universal-time "f_get_un.htm" "get-universal-time")
(getf "f_getf.htm" "getf")
(ghash "f_gethas.htm" "ghash")
(go "s_go.htm" "go")
(graphic-char-p "f_graphi.htm" "graphic-char-p")
(handler-bind "m_handle.htm" "handler-bind")
(handler-case "m_hand_1.htm" "handler-case")
(hash-table "t_hash_t.htm" "hash-table")
(hash-table-count "f_hash_1.htm" "hash-table-count")

```

```

(hash-table-p "f_hash_t.htm" "hash-table-p")
(hash-table-rehash-size "f_hash_2.htm" "hash-table-rehash-size")
(hash-table-rehash-threshold "f_hash_3.htm" "hash-table-rehash-threshold")
(hash-table-size "f_hash_4.htm" "hash-table-size")
(hash-table-test "f_hash_5.htm" "hash-table-test")
(host-namestring "f_namest.htm" "host-namestring")
(identity "f_identi.htm" "identity")
(if "s_if.htm" "if")
(ignorable "d_ignore.htm" "ignorable")
(ignore "d_ignore.htm" "ignore")
(ignore-errors "m_ignore.htm" "ignore-errors")
(imagpart "f_realpa.htm" "imagpart")
(import "f_import.htm" "import")
(in-package "m_in_pkg.htm" "in-package")
(incf "m_incf_.htm" "incf")
(initialize-instance "f_init_i.htm" "initialize-instance")
(inline "d_inline.htm" "inline")
(input-stream-p "f_in_stm.htm" "input-stream-p")
(inspect "f_inspec.htm" "inspect")
(integer "t_intege.htm" "integer")
(integer-decode-float "f_dec_fl.htm" "integer-decode-float")
(integer-length "f_intege.htm" "integer-length")
(integerp "f_inte_1.htm" "integerp")
(interactive-stream-p "f_intera.htm" "interactive-stream-p")
(intern "f_intern.htm" "intern")
(internal-time-units-per-second "v_intern.htm" "internal-time-units-per-second")
(intersection "f_isec_.htm" "intersection")
(invalid-method-error "f_invali.htm" "invalid-method-error")
(invoke-debugger "f_invoke.htm" "invoke-debugger")
(invoke-restart "f_invo_1.htm" "invoke-restart")
(invoke-restart-interactively "f_invo_2.htm" "invoke-restart-interactively")
(isqrt "f_sqrt_.htm" "isqrt")
(keyword "t_kwd.htm" "keyword")
(keywordp "f_kwdp.htm" "keywordp")
(labels "s_flet_.htm" "labels")
(lambda "a_lambda.htm" "lambda")
(lambda-list-keywords "v_lambda.htm" "lambda-list-keywords")
(lambda-parameters-limit "v_lamb_1.htm" "lambda-parameters-limit")
(last "f_last.htm" "last")
(lcm "f_lcm.htm" "lcm")
(ldb "f_ldb.htm" "ldb")
(ldb-test "f_ldb_te.htm" "ldb-test")
(ldiff "f_ldiffc.htm" "ldiff")
(least-negative-double-float "v_most_1.htm" "least-negative-double-float")
(least-negative-long-float "v_most_1.htm" "least-negative-long-float")
(least-negative-normalized-double-float "v_most_1.htm" "least-negative-normalized-double-float")
(least-negative-normalized-long-float "v_most_1.htm" "least-negative-normalized-long-float")
(least-negative-normalized-short-float "v_most_1.htm" "least-negative-normalized-short-float")
(least-negative-normalized-single-float "v_most_1.htm" "least-negative-normalized-single-float")
(least-negative-short-float "v_most_1.htm" "least-negative-short-float")
(least-negative-single-float "v_most_1.htm" "least-negative-single-float")
(least-positive-double-float "v_most_1.htm" "least-positive-double-float")
(least-positive-long-float "v_most_1.htm" "least-positive-long-float")
(least-positive-normalized-double-float "v_most_1.htm" "least-positive-normalized-double-float")
(least-positive-normalized-long-float "v_most_1.htm" "least-positive-normalized-long-float")
(least-positive-normalized-short-float "v_most_1.htm" "least-positive-normalized-short-float")
(least-positive-normalized-single-float "v_most_1.htm" "least-positive-normalized-single-float")
(least-positive-short-float "v_most_1.htm" "least-positive-short-float")
(least-positive-single-float "v_most_1.htm" "least-positive-single-float")
(length "f_length.htm" "length")
(let "s_let_1.htm" "let")
(let* "s_let_1.htm" "letST")
(lisp-implementation-type "f_lisp_i.htm" "lisp-implementation-type")
(lisp-implementation-version "f_lisp_i.htm" "lisp-implementation-version")
(list "a_list.htm" "list")
(list* "f_list_.htm" "listST")
(list-all-packages "f_list_a.htm" "list-all-packages")
(list-length "f_list_1.htm" "list-length")
(listen "f_listen.htm" "listen")
(listp "f_listp.htm" "listp")
(load "f_load.htm" "load")
(load-logical-pathname-translations "f_ld_log.htm" "load-logical-pathname-translations")
(load-time-value "s_ld_tim.htm" "load-time-value")
(locally "s_locall.htm" "locally")
(log "f_log.htm" "log")
(logand "f_logand.htm" "logand")
(logandc1 "f_logand.htm" "logandc1")
(logandc2 "f_logand.htm" "logandc2")
(logbitp "f_logbtp.htm" "logbitp")
(logcount "f_logcou.htm" "logcount")
(logeqv "f_logand.htm" "logeqv")
(logical-pathname "a_logica.htm" "logical-pathname")
(logical-pathname-translations "f_logica.htm" "logical-pathname-translations")
(logior "f_logand.htm" "logior")
(lognand "f_logand.htm" "lognand")
(lognor "f_logand.htm" "lognor")
(lognot "f_logand.htm" "lognot")

```

```

("logorc1" "f_logand.htm" "logorc1")
("logorc2" "f_logand.htm" "logorc2")
("logtest" "f_logtes.htm" "logtest")
("logxor" "f_logand.htm" "logxor")
("long-float" "t_short_.htm" "long-float")
("long-float-epsilon" "v_short_.htm" "long-float-epsilon")
("long-float-negative-epsilon" "v_short_.htm" "long-float-negative-epsilon")
("long-site-name" "f_short_.htm" "long-site-name")
("loop" "m_loop.htm" "loop")
("loop-finish" "m_loop_f.htm" "loop-finish")
("lower-case-p" "f_upper_.htm" "lower-case-p")
("machine-instance" "f_mach_i.htm" "machine-instance")
("machine-type" "f_mach_t.htm" "machine-type")
("machine-version" "f_mach_v.htm" "machine-version")
("macro-function" "f_macro_.htm" "macro-function")
("macroexpand" "f_mexp_.htm" "macroexpand")
("macroexpand-1" "f_mexp_.htm" "macroexpand-1")
("macrolet" "s_flet_.htm" "macrolet")
("make-array" "f_mk_ar.htm" "make-array")
("make-broadcast-stream" "f_mk_bro.htm" "make-broadcast-stream")
("make-concatenated-stream" "f_mk_con.htm" "make-concatenated-stream")
("make-condition" "f_mk_cnd.htm" "make-condition")
("make-dispatch-macro-character" "f_mk_dis.htm" "make-dispatch-macro-character")
("make-echo-stream" "f_mk_ech.htm" "make-echo-stream")
("make-hash-table" "f_mk_has.htm" "make-hash-table")
("make-instance" "f_mk_ins.htm" "make-instance")
("make-instances-obsolete" "f_mk_i_1.htm" "make-instances-obsolete")
("make-list" "f_mk_lis.htm" "make-list")
("make-load-form" "f_mk_ld_.htm" "make-load-form")
("make-load-form-saving-slots" "f_mk_l_1.htm" "make-load-form-saving-slots")
("make-method" "m_call_m.htm" "make-method")
("make-package" "f_mk_pkg.htm" "make-package")
("make-pathname" "f_mk_pn.htm" "make-pathname")
("make-random-state" "f_mk_rnd.htm" "make-random-state")
("make-sequence" "f_mk_seq.htm" "make-sequence")
("make-string" "f_mk_stg.htm" "make-string")
("make-string-input-stream" "f_mk_s_1.htm" "make-string-input-stream")
("make-string-output-stream" "f_mk_s_2.htm" "make-string-output-stream")
("make-symbol" "f_mk_sym.htm" "make-symbol")
("make-synonym-stream" "f_mk_syn.htm" "make-synonym-stream")
("make-two-way-stream" "f_mk_two.htm" "make-two-way-stream")
("makunbound" "f_makunb.htm" "makunbound")
("map" "f_map.htm" "map")
("map-into" "f_map_in.htm" "map-into")
("mapc" "f_mapc_.htm" "mapc")
("mapcan" "f_mapc_.htm" "mapcan")
("mapcar" "f_mapc_.htm" "mapcar")
("mapcon" "f_mapc_.htm" "mapcon")
("maphash" "f_maphas.htm" "maphash")
("mapl" "f_mapc_.htm" "mapl")
("maplist" "f_mapc_.htm" "maplist")
("mask-field" "f_mask_f.htm" "mask-field")
("max" "f_max_m.htm" "max")
("member" "a_member.htm" "member")
("member-if" "f_mem_m.htm" "member-if")
("member-if-not" "f_mem_m.htm" "member-if-not")
("merge" "f_merge.htm" "merge")
("merge-pathnames" "f_merge_.htm" "merge-pathnames")
("method" "t_method.htm" "method")
("method-combination" "a_method.htm" "method-combination")
("method-combination-error" "f_meth_1.htm" "method-combination-error")
("method-qualifiers" "f_method.htm" "method-qualifiers")
("min" "f_max_m.htm" "min")
("minusp" "f_minusp.htm" "minusp")
("mismatch" "f_mismatch.htm" "mismatch")
("mod" "a_mod.htm" "mod")
("most-negative-double-float" "v_most_1.htm" "most-negative-double-float")
("most-negative-fixnum" "v_most_p.htm" "most-negative-fixnum")
("most-negative-long-float" "v_most_1.htm" "most-negative-long-float")
("most-negative-short-float" "v_most_1.htm" "most-negative-short-float")
("most-negative-single-float" "v_most_1.htm" "most-negative-single-float")
("most-positive-double-float" "v_most_1.htm" "most-positive-double-float")
("most-positive-fixnum" "v_most_p.htm" "most-positive-fixnum")
("most-positive-long-float" "v_most_1.htm" "most-positive-long-float")
("most-positive-short-float" "v_most_1.htm" "most-positive-short-float")
("most-positive-single-float" "v_most_1.htm" "most-positive-single-float")
("muffle-warning" "a_muffle.htm" "muffle-warning")
("multiple-value-bind" "m_multip.htm" "multiple-value-bind")
("multiple-value-call" "s_multip.htm" "multiple-value-call")
("multiple-value-list" "m_mult_1.htm" "multiple-value-list")
("multiple-value-prog1" "s_mult_1.htm" "multiple-value-prog1")
("multiple-value-setq" "m_mult_2.htm" "multiple-value-setq")
("multiple-values-limit" "v_multip.htm" "multiple-values-limit")
("name-char" "f_name_c.htm" "name-char")
("namestring" "f_namest.htm" "namestring")
("nbutlast" "f_butlas.htm" "nbutlast")
("nconc" "f_nconc.htm" "nconc")

```

```

("next-method-p" "f_next_m.htm" "next-method-p")
("nil" "a_nil.htm" "nil")
("nintersection" "f_isec_.htm" "nintersection")
("ninth" "f_firstc.htm" "ninth")
("no-applicable-method" "f_no_app.htm" "no-applicable-method")
("no-next-method" "f_no_nex.htm" "no-next-method")
("not" "a_not.htm" "not")
("notany" "f_everyc.htm" "notany")
("notevery" "f_everyc.htm" "notevery")
("notinline" "d_inline.htm" "notinline")
("nreconc" "f_revapp.htm" "nreconc")
("nreverse" "f_revers.htm" "nreverse")
("nset-difference" "f_set_di.htm" "nset-difference")
("nset-exclusive-or" "f_set_ex.htm" "nset-exclusive-or")
("nstring-capitalize" "f_stg_up.htm" "nstring-capitalize")
("nstring-downcase" "f_stg_up.htm" "nstring-downcase")
("nstring-upcase" "f_stg_up.htm" "nstring-upcase")
("nsublis" "f_sublis.htm" "nsublis")
("nsubst" "f_substc.htm" "nsubst")
("nsubst-if" "f_substc.htm" "nsubst-if")
("nsubst-if-not" "f_substc.htm" "nsubst-if-not")
("nsubstitute" "f_sbs_s.htm" "nsubstitute")
("nsubstitute-if" "f_sbs_s.htm" "nsubstitute-if")
("nsubstitute-if-not" "f_sbs_s.htm" "nsubstitute-if-not")
("nth" "f_nth.htm" "nth")
("nth-value" "m_nth_va.htm" "nth-value")
("nthcdr" "f_nthcdr.htm" "nthcdr")
("null" "a_null.htm" "null")
("number" "t_number.htm" "number")
("numberp" "f_nump.htm" "numberp")
("numerator" "f_numera.htm" "numerator")
("nunion" "f_unionc.htm" "nunion")
("oddp" "f_evenpc.htm" "oddp")
("open" "f_open.htm" "open")
("open-stream-p" "f_open_s.htm" "open-stream-p")
("optimize" "d_optimi.htm" "optimize")
("or" "a_or.htm" "or")
("otherwise" "m_case_.htm" "otherwise")
("output-stream-p" "f_in_stm.htm" "output-stream-p")
("package" "t_pkg.htm" "package")
("package-error" "e_pkg_er.htm" "package-error")
("package-error-package" "f_pkg_er.htm" "package-error-package")
("package-name" "f_pkg_na.htm" "package-name")
("package-nicknames" "f_pkg_ni.htm" "package-nicknames")
("package-shadowing-symbols" "f_pkg_sh.htm" "package-shadowing-symbols")
("package-use-list" "f_pkg_us.htm" "package-use-list")
("package-used-by-list" "f_pkg__l.htm" "package-used-by-list")
("packagep" "f_pkgp.htm" "packagep")
("pairlis" "f_pairli.htm" "pairlis")
("parse-error" "e_parse_.htm" "parse-error")
("parse-integer" "f_parse_.htm" "parse-integer")
("parse-namestring" "f_pars_l.htm" "parse-namestring")
("pathname" "a_pn.htm" "pathname")
("pathname-device" "f_pn_hos.htm" "pathname-device")
("pathname-directory" "f_pn_hos.htm" "pathname-directory")
("pathname-host" "f_pn_hos.htm" "pathname-host")
("pathname-match-p" "f_pn_mat.htm" "pathname-match-p")
("pathname-name" "f_pn_hos.htm" "pathname-name")
("pathname-type" "f_pn_hos.htm" "pathname-type")
("pathname-version" "f_pn_hos.htm" "pathname-version")
("pathnamep" "f_pnp.htm" "pathnamep")
("peek-char" "f_peek_c.htm" "peek-char")
("phase" "f_phase.htm" "phase")
("pi" "v_pi.htm" "pi")
("plusp" "f_minusp.htm" "plusp")
("pop" "m_pop.htm" "pop")
("position" "f_pos_p.htm" "position")
("position-if" "f_pos_p.htm" "position-if")
("position-if-not" "f_pos_p.htm" "position-if-not")
("pprint" "f_wr_pr.htm" "pprint")
("pprint-dispatch" "f_ppr_di.htm" "pprint-dispatch")
("pprint-exit-if-list-exhausted" "m_ppr_ex.htm" "pprint-exit-if-list-exhausted")
("pprint-fill" "f_ppr_fi.htm" "pprint-fill")
("pprint-indent" "f_ppr_in.htm" "pprint-indent")
("pprint-linear" "f_ppr_fi.htm" "pprint-linear")
("pprint-logical-block" "m_ppr_lo.htm" "pprint-logical-block")
("pprint-newline" "f_ppr_nl.htm" "pprint-newline")
("pprint-pop" "m_ppr_po.htm" "pprint-pop")
("pprint-tab" "f_ppr_ta.htm" "pprint-tab")
("pprint-tabular" "f_ppr_fi.htm" "pprint-tabular")
("prinl" "f_wr_pr.htm" "prinl")
("prinl-to-string" "f_wr_to_.htm" "prinl-to-string")
("princ" "f_wr_pr.htm" "princ")
("princ-to-string" "f_wr_to_.htm" "princ-to-string")
("print" "f_wr_pr.htm" "print")
("print-not-readable" "e_pr_not.htm" "print-not-readable")
("print-not-readable-object" "f_pr_not.htm" "print-not-readable-object")

```

```

(print-object "f_pr_obj.htm" "print-object")
(print-unreadable-object "m_pr_unr.htm" "print-unreadable-object")
(probe-file "f_probe_.htm" "probe-file")
(proclaim "f_procla.htm" "proclaim")
(prog "m_prog_.htm" "prog")
(prog* "m_prog_.htm" "prog*")
(prog1 "m_proglc.htm" "prog1")
(prog2 "m_proglc.htm" "prog2")
(progn "s_progn.htm" "progn")
(program-error "e_progra.htm" "program-error")
(progv "s_progv.htm" "progv")
(provide "f_provid.htm" "provide")
(psetf "m_setf_.htm" "psetf")
(psetq "m_psetq.htm" "psetq")
(push "m_push.htm" "push")
(pushnew "m_pshnew.htm" "pushnew")
(quote "s_quote.htm" "quote")
(random "f_random.htm" "random")
(random-state "t_rnd_st.htm" "random-state")
(random-state-p "f_rnd_st.htm" "random-state-p")
(rassoc "f_rassoc.htm" "rassoc")
(rassoc-if "f_rassoc.htm" "rassoc-if")
(rassoc-if-not "f_rassoc.htm" "rassoc-if-not")
(ratio "t_ratio.htm" "ratio")
(rational "a_ration.htm" "rational")
(rationalize "f_ration.htm" "rationalize")
(rationalp "f_rati_l.htm" "rationalp")
(read "f_rd_rd.htm" "read")
(read-byte "f_rd_by.htm" "read-byte")
(read-char "f_rd_cha.htm" "read-char")
(read-char-no-hang "f_rd_c_l.htm" "read-char-no-hang")
(read-delimited-list "f_rd_del.htm" "read-delimited-list")
(read-from-string "f_rd_fro.htm" "read-from-string")
(read-line "f_rd_lin.htm" "read-line")
(read-preserving-whitespace "f_rd_rd.htm" "read-preserving-whitespace")
(read-sequence "f_rd_seq.htm" "read-sequence")
(reader-error "e_rder_e.htm" "reader-error")
(readtable "t_rdtabl.htm" "readtable")
(readtable-case "f_rdtabl.htm" "readtable-case")
(readtablep "f_rdtal_l.htm" "readtablep")
(real "t_real.htm" "real")
(realp "f_realp.htm" "realp")
(realpart "f_realpa.htm" "realpart")
(reduce "f_reduce.htm" "reduce")
(reinitialize-instance "f_reinit.htm" "reinitialize-instance")
(rem "f_mod_r.htm" "rem")
(remf "m_remf.htm" "remf")
(remhash "f_remhas.htm" "remhash")
(remove "f_rm_rm.htm" "remove")
(remove-duplicates "f_rm_dup.htm" "remove-duplicates")
(remove-if "f_rm_rm.htm" "remove-if")
(remove-if-not "f_rm_rm.htm" "remove-if-not")
(remove-method "f_rm_met.htm" "remove-method")
(remprop "f_rempro.htm" "remprop")
(rename-file "f_rn_fil.htm" "rename-file")
(rename-package "f_rn_pkg.htm" "rename-package")
(replace "f_replac.htm" "replace")
(require "f_provid.htm" "require")
(rest "f_rest.htm" "rest")
(restart "t_rst.htm" "restart")
(restart-bind "m_rst_bi.htm" "restart-bind")
(restart-case "m_rst_ca.htm" "restart-case")
(restart-name "f_rst_na.htm" "restart-name")
(return "m_return.htm" "return")
(return-from "s_ret_fr.htm" "return-from")
(revappend "f_revapp.htm" "revappend")
(reverse "f_revers.htm" "reverse")
(room "f_room.htm" "room")
(rotatef "m_rotate.htm" "rotatef")
(round "f_floorc.htm" "round")
(row-major-aref "f_row_ma.htm" "row-major-aref")
(rplaca "f_rplaca.htm" "rplaca")
(rplacd "f_rplaca.htm" "rplacd")
(safety "d_optimi.htm" "safety")
(satisfies "t_satisf.htm" "satisfies")
(sbit "f_bt_sb.htm" "sbit")
(scale-float "f_dec_fl.htm" "scale-float")
(schar "f_char_.htm" "schar")
(search "f_search.htm" "search")
(second "f_firstc.htm" "second")
(sequence "t_seq.htm" "sequence")
(serious-condition "e_seriou.htm" "serious-condition")
(set "f_set.htm" "set")
(set-difference "f_set_di.htm" "set-difference")
(set-dispatch-macro-character "f_set_l.htm" "set-dispatch-macro-character")
(set-exclusive-or "f_set_ex.htm" "set-exclusive-or")
(set-macro-character "f_set_ma.htm" "set-macro-character")

```

```

("set-pprint-dispatch" "f_set_pp.htm" "set-pprint-dispatch")
("set-syntax-from-char" "f_set_sy.htm" "set-syntax-from-char")
("setf" "a_setf.htm" "setf")
("setq" "s_setq.htm" "setq")
("seventh" "f_firstc.htm" "seventh")
("shadow" "f_shadow.htm" "shadow")
("shadowing-import" "f_shdw_i.htm" "shadowing-import")
("shared-initialize" "f_shared.htm" "shared-initialize")
("shiftf" "m_shiftf.htm" "shiftf")
("short-float" "t_short_.htm" "short-float")
("short-float-epsilon" "v_short_.htm" "short-float-epsilon")
("short-float-negative-epsilon" "v_short_.htm" "short-float-negative-epsilon")
("short-site-name" "f_short_.htm" "short-site-name")
("signal" "f_signal.htm" "signal")
("signed-byte" "t_sgn_by.htm" "signed-byte")
("signum" "f_signum.htm" "signum")
("simple-array" "t_smp_ar.htm" "simple-array")
("simple-base-string" "t_smp_ba.htm" "simple-base-string")
("simple-bit-vector" "t_smp_bt.htm" "simple-bit-vector")
("simple-bit-vector-p" "f_smp_bt.htm" "simple-bit-vector-p")
("simple-condition" "e_smp_cn.htm" "simple-condition")
("simple-condition-format-arguments" "f_smp_cn.htm" "simple-condition-format-arguments")
("simple-condition-format-control" "f_smp_cn.htm" "simple-condition-format-control")
("simple-error" "e_smp_er.htm" "simple-error")
("simple-string" "t_smp_st.htm" "simple-string")
("simple-string-p" "f_smp_st.htm" "simple-string-p")
("simple-type-error" "e_smp_tp.htm" "simple-type-error")
("simple-vector" "t_smp_ve.htm" "simple-vector")
("simple-vector-p" "f_smp_ve.htm" "simple-vector-p")
("simple-warning" "e_smp_wa.htm" "simple-warning")
("sin" "f_sin_c.htm" "sin")
("single-float" "t_short_.htm" "single-float")
("single-float-epsilon" "v_short_.htm" "single-float-epsilon")
("single-float-negative-epsilon" "v_short_.htm" "single-float-negative-epsilon")
("sinh" "f_sinh_.htm" "sinh")
("sixth" "f_firstc.htm" "sixth")
("sleep" "f_sleep.htm" "sleep")
("slot-boundp" "f_slt_bo.htm" "slot-boundp")
("slot-exists-p" "f_slt_ex.htm" "slot-exists-p")
("slot-makunbound" "f_slt_ma.htm" "slot-makunbound")
("slot-missing" "f_slt_mi.htm" "slot-missing")
("slot-unbound" "f_slt_un.htm" "slot-unbound")
("slot-value" "f_slt_va.htm" "slot-value")
("software-type" "f_sw_tpc.htm" "software-type")
("software-version" "f_sw_tpc.htm" "software-version")
("some" "f_everyc.htm" "some")
("sort" "f_sort_.htm" "sort")
("space" "d_optimi.htm" "space")
("special" "d_specia.htm" "special")
("special-operator-p" "f_specia.htm" "special-operator-p")
("speed" "d_optimi.htm" "speed")
("sqrt" "f_sqrt_.htm" "sqrt")
("stable-sort" "f_sort_.htm" "stable-sort")
("standard" "07_ffb.htm" "standard")
("standard-char" "t_std_ch.htm" "standard-char")
("standard-char-p" "f_std_ch.htm" "standard-char-p")
("standard-class" "t_std_cl.htm" "standard-class")
("standard-generic-function" "t_std_ge.htm" "standard-generic-function")
("standard-method" "t_std_me.htm" "standard-method")
("standard-object" "t_std_ob.htm" "standard-object")
("step" "m_step.htm" "step")
("storage-condition" "e_storag.htm" "storage-condition")
("store-value" "a_store_.htm" "store-value")
("stream" "t_stream.htm" "stream")
("stream-element-type" "f_stm_el.htm" "stream-element-type")
("stream-error" "e_stm_er.htm" "stream-error")
("stream-error-stream" "f_stm_er.htm" "stream-error-stream")
("stream-external-format" "f_stm_ex.htm" "stream-external-format")
("streamp" "f_stmp.htm" "streamp")
("string" "a_string.htm" "string")
("string-capitalize" "f_stg_up.htm" "string-capitalize")
("string-downcase" "f_stg_up.htm" "string-downcase")
("string-equal" "f_stgeq_.htm" "string-equal")
("string-greaterp" "f_stgeq_.htm" "string-greaterp")
("string-left-trim" "f_stg_tr.htm" "string-left-trim")
("string-lessp" "f_stgeq_.htm" "string-lessp")
("string-not-equal" "f_stgeq_.htm" "string-not-equal")
("string-not-greaterp" "f_stgeq_.htm" "string-not-greaterp")
("string-not-lessp" "f_stgeq_.htm" "string-not-lessp")
("string-right-trim" "f_stg_tr.htm" "string-right-trim")
("string-stream" "t_stg_st.htm" "string-stream")
("string-trim" "f_stg_tr.htm" "string-trim")
("string-upcase" "f_stg_up.htm" "string-upcase")
("string/= " "f_stgeq_.htm" "stringSLEQ")
("string< " "f_stgeq_.htm" "stringLT")
("string<=" "f_stgeq_.htm" "stringLTEQ")
("string=" "f_stgeq_.htm" "stringEQ")

```

```

("string>" "f_stgeq_.htm" "stringGT")
("string>=" "f_stgeq_.htm" "stringGTEQ")
("stringp" "f_stgp.htm" "stringp")
("structure" "f_docume.htm" "structure")
("structure-class" "t_stu_cl.htm" "structure-class")
("structure-object" "t_stu_ob.htm" "structure-object")
("style-warning" "e_style_.htm" "style-warning")
("sublis" "f_sublis.htm" "sublis")
("subseq" "f_subseq.htm" "subseq")
("subsetp" "f_subset.htm" "subsetp")
("subst" "f_substc.htm" "subst")
("subst-if" "f_substc.htm" "subst-if")
("subst-if-not" "f_substc.htm" "subst-if-not")
("substitute" "f_sbs_s.htm" "substitute")
("substitute-if" "f_sbs_s.htm" "substitute-if")
("substitute-if-not" "f_sbs_s.htm" "substitute-if-not")
("subtypep" "f_subtpp.htm" "subtypep")
("svref" "f_svref.htm" "svref")
("sxhash" "f_sxhash.htm" "sxhash")
("symbol" "t_symbol.htm" "symbol")
("symbol-function" "f_symb_1.htm" "symbol-function")
("symbol-macrolet" "s_symbol.htm" "symbol-macrolet")
("symbol-name" "f_symb_2.htm" "symbol-name")
("symbol-package" "f_symb_3.htm" "symbol-package")
("symbol-plist" "f_symb_4.htm" "symbol-plist")
("symbol-value" "f_symb_5.htm" "symbol-value")
("symbolp" "f_symbol.htm" "symbolp")
("synonym-stream" "t_syn_st.htm" "synonym-stream")
("synonym-stream-symbol" "f_syn_st.htm" "synonym-stream-symbol")
("t" "a_t.htm" "t")
("tagbody" "s_tagbod.htm" "tagbody")
("tailp" "f_ldifffc.htm" "tailp")
("tan" "f_sin_c.htm" "tan")
("tanh" "f_sinh_.htm" "tanh")
("tenth" "f_firstc.htm" "tenth")
("terpri" "f_terpri.htm" "terpri")
("the" "s_the.htm" "the")
("third" "f_firstc.htm" "third")
("throw" "s_throw.htm" "throw")
("time" "m_time.htm" "time")
("trace" "m_tracec.htm" "trace")
("translate-logical-pathname" "f_tr_log.htm" "translate-logical-pathname")
("translate-pathname" "f_tr_pn.htm" "translate-pathname")
("tree-equal" "f_tree_e.htm" "tree-equal")
("truename" "f_tn.htm" "truename")
("truncate" "f_floorc.htm" "truncate")
("two-way-stream" "t_two_wa.htm" "two-way-stream")
("two-way-stream-input-stream" "f_two_wa.htm" "two-way-stream-input-stream")
("two-way-stream-output-stream" "f_two_wa.htm" "two-way-stream-output-stream")
("type" "a_type.htm" "type")
("type-error" "e_tp_err.htm" "type-error")
("type-error-datum" "f_tp_err.htm" "type-error-datum")
("type-error-expected-type" "f_tp_err.htm" "type-error-expected-type")
("type-of" "f_tp_of.htm" "type-of")
("typecase" "m_tpcase.htm" "typecase")
("typep" "f_typep.htm" "typep")
("unbound-slot" "e_unboun.htm" "unbound-slot")
("unbound-slot-instance" "f_unboun.htm" "unbound-slot-instance")
("unbound-variable" "e_unbo_1.htm" "unbound-variable")
("undefined-function" "e_undefi.htm" "undefined-function")
("unexport" "f_unexpo.htm" "unexport")
("unintern" "f_uninte.htm" "unintern")
("union" "f_uni onc.htm" "union")
("unless" "m_when_.htm" "unless")
("unread-char" "f_unrd_c.htm" "unread-char")
("unsigned-byte" "t_unsgn_.htm" "unsigned-byte")
("untrace" "m_tracec.htm" "untrace")
("unuse-package" "f_unuse_.htm" "unuse-package")
("unwind-protect" "s_unwind.htm" "unwind-protect")
("update-instance-for-different-class" "f_update.htm" "update-instance-for-different-class")
("update-instance-for-redefined-class" "f_upda_1.htm" "update-instance-for-redefined-class")
("upgraded-array-element-type" "f_upgr_1.htm" "upgraded-array-element-type")
("upgraded-complex-part-type" "f_upgrad.htm" "upgraded-complex-part-type")
("upper-case-p" "f_upper_.htm" "upper-case-p")
("use-package" "f_use_pk.htm" "use-package")
("use-value" "a_use_va.htm" "use-value")
("user-homedir-pathname" "f_user_h.htm" "user-homedir-pathname")
("values" "a_values.htm" "values")
("values-list" "f_vals_l.htm" "values-list")
("variable" "f_docume.htm" "variable")
("vector" "a_vector.htm" "vector")
("vector-pop" "f_vec_po.htm" "vector-pop")
("vector-push" "f_vec_ps.htm" "vector-push")
("vector-push-extend" "f_vec_ps.htm" "vector-push-extend")
("vectorp" "f_vecp.htm" "vectorp")
("warn" "f_warn.htm" "warn")
("warning" "e_warnin.htm" "warning")

```

```

("when" "m_when_.htm" "when")
("wild-pathname-p" "f_wild_p.htm" "wild-pathname-p")
("with-accessors" "m_w_acce.htm" "with-accessors")
("with-compilation-unit" "m_w_comp.htm" "with-compilation-unit")
("with-condition-restarts" "m_w_cnd_.htm" "with-condition-restarts")
("with-hash-table-iterator" "m_w_hash.htm" "with-hash-table-iterator")
("with-input-from-string" "m_w_in_f.htm" "with-input-from-string")
("with-open-file" "m_w_open.htm" "with-open-file")
("with-open-stream" "m_w_op_l.htm" "with-open-stream")
("with-output-to-string" "m_w_out_.htm" "with-output-to-string")
("with-package-iterator" "m_w_pkg_.htm" "with-package-iterator")
("with-simple-restart" "m_w_smp_.htm" "with-simple-restart")
("with-slots" "m_w_slts.htm" "with-slots")
("with-standard-io-syntax" "m_w_std_.htm" "with-standard-io-syntax")
("write" "f_wr_pr.htm" "write")
("write-byte" "f_wr_by.htm" "write-byte")
("write-char" "f_wr_cha.htm" "write-char")
("write-line" "f_wr_stg.htm" "write-line")
("write-sequence" "f_wr_seq.htm" "write-sequence")
("write-string" "f_wr_stg.htm" "write-string")
("write-to-string" "f_wr_to_.htm" "write-to-string")
("y-or-n-p" "f_y_or_n.htm" "y-or-n-p")
("yes-or-no-p" "f_y_or_n.htm" "yes-or-no-p")
("zerop" "f_zerop.htm" "zerop"))

```

```
(RPAQ? CLHS.ROOT.URL "https://interlisp.org/clhs/")
```

```
(RPAQ? CLHS.INDEX )
```

```
(RPAQ? CLHS.OPENER )
```

```
(RPAQ? HELPSYS.REPO.TYPES ' (FNS FUNCTIONS VARS VARIABLES))
```

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY
```

```
(GLOBALVARS CLHS.INDEX CLHS.OPENER HELPSYS.REPO.TYPES CLHS.ROOT.URL)
)
```

;;; Interface to DInfo

```
(DEFINEQ
```

(IRM.GET.DINFOGRAPH

```
[LAMBDA (FROM.BACKGROUND?)
```

; Edited 14-Aug-87 17:31 by drc:

;; returns the DInfo graph for the IRM, ensuring that it has been setup.

```
(CL:UNLESS (TYPEP IRM.DINFOGRAPH 'DINFOGRAPH)
```

;; graph has not been loaded -- load it

```
(RESETFORM (TTYDISPLAYSTREAM PROMPTWINDOW)
  (SETQ IRM.DINFOGRAPH (IRM.LOAD-GRAPH)))
```

```
(CL:UNLESS (WINDOWP (fetch (DINFOGRAPH WINDOW) of IRM.DINFOGRAPH))
```

;; graph has not been set up -- set it up

```
(DINFO IRM.DINFOGRAPH (CREATEW (GETBOXREGION 540 400 NIL NIL NIL "Specify region for IRM DInfo window")
  "IRM DInfo Graph")
```

```
T
  (NOT FROM.BACKGROUND?)))
```

```
IRM.DINFOGRAPH])
```

(IRM.DISPLAY.REF

```
[LAMBDA (REF GRAPH)
```

; Edited 19-Aug-2022 20:21 by lmm
(* drc%: "18-Jan-86 17:17")

;;; visit the DInfo node of GRAPH containing REF

```
(LET [(NODE (FASSOC (fetch (IRMREFERENCE NODE) of REF)
  (fetch (DINFOGRAPH NODELST) of GRAPH]
```

```
(if NODE
  then (DINFO.UPDATE GRAPH NODE (LIST (fetch (IRMREFERENCE ITEM) of REF)
    (fetch (IRMREFERENCE CH#) of REF]))
```

)

(CL:DEFUN IRM.LOAD-GRAPH ()

```
[LET [(FILE (INFILEP (PACKFILENAME 'NAME 'IRM 'EXTENSION 'DINFOGRAPH 'BODY IRM.HOST&DIR)
  (CL:IF FILE
```

```
(DINFO.READ.GRAPH FILE)
  (PROG1 NIL (CL:WARN "IRM.DINFOGRAPH not found on ~S~%Perhaps IL:IRM.HOST&DIR is set incorrectly"
    IRM.HOST&DIR)))]])
```

```
(ADDTOVAR DINFO.GRAPHS ("Interlisp-D Reference Manual" (IRM.GET.DINFOGRAPH T)))
```

```
(RPAQ? IRM.DINFOGRAPH )
```


(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS IRM.DINFOGRAPH)
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

[COND
 (IRM.HOST&DIR (SETQ IRM.DINFOGRAPH (IRM.LOAD-GRAPH)
)

;;; Cross reference imageobj

(DEFINEQ

(IRM.DISPLAY.CREF

(* drc%: " 7-Jan-86 13:41")

```
[LAMBDA (IMAGEOBJ STREAM)
  (if (EQ (IMAGESTREAMTYPE STREAM)
        'DISPLAY)
      then (DSPFONT IRM.CREF.FONT STREAM)
          (LET* ((STRING (IMAGEOBJPROP IMAGEOBJ 'ITEM))
                (STRINGREGION (STRINGREGION STRING STREAM))
                (LEFT (ADD1 (fetch (REGION LEFT) of STRINGREGION)))
                (BOTTOM (fetch (REGION BOTTOM) of STRINGREGION))
                (REGION (create REGION
                                LEFT _ LEFT
                                BOTTOM _ BOTTOM
                                HEIGHT _ (IPLUS (fetch (REGION HEIGHT) of STRINGREGION)
                                                2)
                                WIDTH _ (IPLUS (fetch (REGION WIDTH) of STRINGREGION)
                                                6)))
                (TOP (fetch (REGION TOP) of REGION))
                (RIGHT (fetch (REGION RIGHT) of REGION))
                (IMAGEOBJPROP IMAGEOBJ 'REGION REGION)
                (CENTERPRINTINREGION STRING REGION STREAM)
                (DRAWLINE LEFT BOTTOM LEFT (SUB1 TOP)
                          1
                          'INVERT STREAM)
                (DRAWLINE LEFT TOP (SUB1 RIGHT)
                          TOP 1 'INVERT STREAM)
                (DRAWLINE RIGHT TOP RIGHT (ADD1 BOTTOM)
                          1
                          'INVERT STREAM)
                (DRAWLINE RIGHT BOTTOM (ADD1 LEFT)
                          BOTTOM 1 'INVERT STREAM))
          else (PRIN1 "page X.XX" STREAM])
```

(IRM.CREF.BOX

(* drc%: " 7-Jan-86 13:42")

```
[LAMBDA (IMAGEOBJ STREAM CURRENTX RIGHTMARGIN)
  (LET ((TYPE (IMAGESTREAMTYPE STREAM))
        (create IMAGEBOX
                 XSIZE _ (SELECTQ TYPE
                                   (DISPLAY (IPLUS (STRINGWIDTH (IMAGEOBJPROP IMAGEOBJ 'ITEM)
                                                         IRM.CREF.FONT)
                                                   8))
                                   (STRINGWIDTH "page X.XX" STREAM))
                 YSIZE _ (SELECTQ TYPE
                                   (DISPLAY (IPLUS (FONTHEIGHT IRM.CREF.FONT)
                                                   4))
                                   (FONTHEIGHT STREAM))
                 YDESC _ (SELECTQ TYPE
                          (DISPLAY 4)
                          0)
                 XKERN _ 0])
```

(IRM.PUT.CREF

(* drc%: " 7-Jan-86 22:09")

```
[LAMBDA (IMAGEOBJ STREAM)
  (PRIN2 (CONS (IMAGEOBJPROP IMAGEOBJ 'ITEM)
              (IMAGEOBJPROP IMAGEOBJ 'TYPE))
        STREAM])
```

(IRM.GET.CREF

(* drc%: " 2-Jan-86 17:45")

```
[LAMBDA (FILE TEXTSTREAM)
  (DECLARE (GLOBALVARS \IRM.CREF.IMAGEFNS))
  (LET ((DATA (READ FILE))
        (IMAGEOBJ (IMAGEOBJCREATE NIL \IRM.CREF.IMAGEFNS))
        (IMAGEOBJPROP IMAGEOBJ 'ITEM (CAR DATA))
        (IMAGEOBJPROP IMAGEOBJ 'TYPE (CDR DATA))
        IMAGEOBJ])
```

(IRM.CREF.BUTTONEVENTFN

```
[LAMBDA (IMAGEOBJ WSTREAM SELECTION RELX RELY WINDOW TEXTSTREAM BUTTON)
(* drc%: "8-Jan-86 15:34")
(* (INSPECT IMAGEOBJ))

(LET* ((BOUNDBOX (IMAGEOBJPROP IMAGEOBJ 'BOUNDBOX))
(WIDTH (fetch (IMAGEBOX XSIZE) of BOUNDBOX))
(HEIGHT (fetch (IMAGEBOX YSIZE) of BOUNDBOX))
(REGION (create REGION
HEIGHT _ HEIGHT
WIDTH _ WIDTH
LEFT _ 0
BOTTOM _ 0)))
(RESETFORM (TTYDISPLAYSTREAM (GETPROMPTWINDOW WINDOW))
(BLTSHADE BLACKSHADE WSTREAM 0 0 WIDTH HEIGHT 'INVERT)
(bind (N _ 0)
(ITEM _ (IMAGEOBJPROP IMAGEOBJ 'ITEM))
(TYPE _ (IMAGEOBJPROP IMAGEOBJ 'TYPE)) until [OR (NOT (MOUSESTATE (OR LEFT MIDDLE)))
(NOT (INSIDEP REGION (CURSORPOSITION NIL
WSTREAM))

do (BLOCK 100)
(if (EQ (SETQ N (ADD1 N))
10)
then (printout T T "Will lookup " (IMAGEOBJPROP IMAGEOBJ 'ITEM)
(if TYPE
then (CONCAT " as a " TYPE ".")
else ".")))
(GETMOUSESTATE)
finally (CLEARW T)
(if (INSIDEP REGION (CURSORPOSITION NIL WSTREAM))
then (ADD.PROCESS (LIST 'IRM.LOOKUP (KWOTE ITEM)
(KWOTE TYPE)
(WINDOWPROP WINDOW 'DINFOGRAPH))
'NAME "IRM Cross Reference"))
(BLTSHADE BLACKSHADE WSTREAM 0 0 WIDTH HEIGHT 'INVERT)
NIL]))

)
```

```
(RPAQ? IRM.CREF.FONT (FONTCREATE 'MODERN 8 'MRR))
```

```
(RPAQ? \IRM.CREF.IMAGEFNS (IMAGEFNSCREATE (FUNCTION IRM.DISPLAY.CREF)
(FUNCTION IRM.CREF.BOX)
(FUNCTION IRM.PUT.CREF)
(FUNCTION IRM.GET.CREF)
(FUNCTION NIL)
(FUNCTION IRM.CREF.BUTTONEVENTFN)))
```

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY
```

```
(GLOBALVARS IRM.CREF.FONT \IRM.CREF.IMAGEFNS)
)
```

;;; Internal functions and variables

```
(DEFINEQ
```

(IRM.GET.REF

```
[LAMBDA (KEYWORD TYPE)
```

; Edited 19-Aug-2022 20:00 by lmm
(* drc%: "18-Jan-86 17:13")

;;; Returns an IRMREFERENCE for KEYWORD of optionally specified TYPE.

(IRM.GET.HASHFILE)

;; keywords in hashfile are all uppercased -- makes lookup case insensitive;

```
(SETQ KEYWORD (MKATOM (U-CASE KEYWORD)))
(LET ((REFS (GETHASHFILE KEYWORD \IRM.HASHFILE)))
(COND
```

```
((NULL REFS)
NIL)
((NULL TYPE)
(IRM.CHOOSE.REF REFS KEYWORD))
((for REF in REFS thereis (if (AND (EQ (fetch (IRMREFERENCE TYPE) of REF)
TYPE)
(fetch (IRMREFERENCE PRIMARYFLG) of REF))
then REF)))
((SETQ REFS (for REF in REFS join (if (EQ (fetch (IRMREFERENCE TYPE) of REF)
TYPE)
then (LIST REF)
else NIL))))
(IRM.CHOOSE.REF REFS KEYWORD]))
```

(IRM.SMART.REF

```
[LAMBDA (KEYWORD)
```

; Edited 19-Aug-2022 20:46 by lmm
(* drc%: "18-Jan-86 17:40")

;; Returns IRMREFERENCE for KEYWORD. Allows wildcards in KEYWORD, and will try spelling correction.

```
(if (while [SETQ POS (STRPOS "*" KEYWORD (AND POS (ADD1 POS) bind POS when (NEQ (NTHCHAR KEYWORD (SUB1 POS))
'%)
do (RETURN T) finally ; if not doing wildcarding then remove quotes when preceding
; asterisks
[SETQ KEYWORD (PACK (for TAIL on (UNPACK KEYWORD)
when [NOT (AND (EQ (CAR TAIL)
'%)
(EQ (CADR TAIL)
'*]
collect (CAR TAIL)
(RETURN NIL))
then (\IRM.WILD.REF KEYWORD) ; there's an unquoted asterisk -- it's wildcardin' time!
elseif \IRM.KEYWORDS
then [RESETFORM (TTY.PROCESS (THIS.PROCESS)) ; we've got possible matches loaded, so try spelling correction
(LET ((CORRECTED (MISPELLED? KEYWORD 50 \IRM.KEYWORDS T))
(if CORRECTED
then (\IRM.GET.REF CORRECTED]
else (\IRM.GET.REF KEYWORD)]) ; default to normal lookup
```

(\IRM.CHOOSE.REF

```
[LAMBDA (REFS KEYWORD) (* drc%: "8-Jan-86 15:23")
(if (NULL (CDR REFS))
then (CAR REFS)
else (MENU (create MENU
CENTERFLG _ T
TITLE _ (MKSTRING KEYWORD)
ITEMS _ (for REF in REFS collect (LIST (LET ((TYPE (fetch (IRMREFERENCE TYPE) of REF)))
(if (fetch (IRMREFERENCE PRIMARYFLG) of REF)
then (PACK* "*" " TYPE " *)
else TYPE))
(KWOTE REF)
(CONCAT "Lookup " KEYWORD " as " (fetch (IRMREFERENCE
TYPE)
of REF))
```

(\IRM.WILD.REF

```
[LAMBDA (KEYWORD) ; Edited 19-Aug-2022 20:31 by lmm
(* drc%: "18-Jan-86 17:04")
;; Return IRMREFERENCE matching wildcarded KEYWORD.
(LET* ((MATCHES (\IRM.WILDCARD KEYWORD)))
(if MATCHES
then (if (NULL (CDR MATCHES))
then (\IRM.GET.REF (CAR MATCHES))
else (LET [(CHOICE (MENU (create MENU
ITEMS _ (for MATCH in MATCHES
collect (LIST MATCH (KWOTE MATCH)
(CONCAT "Will lookup " MATCH " in IRM
if selected."))
CENTERFLG _ T
TITLE _ KEYWORD]
(AND CHOICE (\IRM.GET.REF CHOICE]))
```

(\IRM.WILDCARD

```
[LAMBDA (WILDATOM LIST) (* drc%: "18-Jan-86 17:00")
(** Returns those atoms in LIST which match WILDATOM.)
(LET ((SCRATCH (CONS)
(WILDLIST (UNPACK WILDATOM)))
(for ATOM in LIST when (\IRM.WILD.MATCH WILDLIST (DUNPACK ATOM SCRATCH)) collect ATOM])
```

(\IRM.WILD.MATCH

```
[LAMBDA (WILDLIST LIST) (* drc%: "18-Jan-86 16:59")
(** predicate for whether wildcard containing WILDLIST matches LIST.)
(COND
((AND (NULL WILDLIST)
(NULL LIST)))
[(AND (EQ (CAR WILDLIST)
'%)
(EQ (CADR WILDLIST)
'*)
(if (EQ '* (CAR LIST))
then (\IRM.WILD.MATCH (CDDR WILDLIST)
(CDR LIST)
(* found a quoted asterisk)
(* and it matches)
```

```

[ (EQ (CAR WILDLIST)
      '* ) (* found a real wildcard)
  (OR (NULL (CDR WILDLIST))
      (for TAIL on LIST thereis (\IRM.WILD.MATCH (CDR WILDLIST)
                                                  TAIL]
      ((EQ (CAR WILDLIST)
           (CAR LIST)) (* first chars match -- keep checking)
        (\IRM.WILD.MATCH (CDR WILDLIST)
                          (CDR LIST)))
      (T NIL])

```

(\IRM.GET.HASHFILE

```

[LAMBDA NIL (* drc%: "16-Dec-85 12:09")
  (OR (ARRAYP \IRM.HASHFILE)
      (SETQ \IRM.HASHFILE (OPENHASHFILE (OR IRM.HASHFILE.NAME (PACKFILENAME 'NAME 'IRM 'EXTENSION 'HASHFILE
                                                                              'BODY IRM.HOST&DIR))
                                          'INPUT]))

```

(\IRM.GET.KEYWORDS

```

[LAMBDA NIL ; Edited 19-Aug-2022 20:33 by lmm
          (* drc%: "18-Jan-86 17:14")

```

;;; keyword list is hidden in hashfile as its key is in lower case

```

  (OR \IRM.KEYWORDS (PROGN (\IRM.GET.HASHFILE)
                            (SETQ \IRM.KEYWORDS (GETHASHFILE 'irm.keywords (\IRM.GET.HASHFILE]))
                          )
    (RPAQ? \IRM.HASHFILE )
    (RPAQ? \IRM.KEYWORDS )
    (DECLARE%: DOEVAL@COMPILE DONTCOPY
    (GLOBALVARS \IRM.HASHFILE \IRM.KEYWORDS)
    )
    (CL:DEFUN \IRM.AROUND-EXIT (EVENT)
      (CASE EVENT
        ((BEFORELOGOUT BEFOREMAKESYS BEFORESYSOUT) (AND \IRM.HASHFILE (CLOSEHASHFILE \IRM.HASHFILE))))
    )
    (ADDTOVAR AROUNDEXITFNS \IRM.AROUND-EXIT)
    (PUTPROPS HELPSYS FILETYPE :FAKE-COMPILE-FILE)

```

FUNCTION INDEX

CLHS.INDEX	3	IRM.DISPLAY.CREF	17	IRM.RESET	3	\IRM.GET.REF	18
CLHS.LOOKUP	4	IRM.DISPLAY.REF	16	IRM.SMART.LOOKUP	3	\IRM.SMART.REF	18
CLHS.OPENER	4	IRM.GET.CREF	17	REPO.LOOKUP	4	\IRM.WILD.MATCH	19
GENERIC.MAN.LOOKUP	2	IRM.GET.DINFOGRAPH	16	\IRM.AROUND-EXIT	20	\IRM.WILD.REF	19
HELPSYS	2	IRM.LOAD-GRAPH	16	\IRM.CHOOSE.REF	19	\IRM.WILDCARD	19
IRM.CREF.BOX	17	IRM.LOOKUP	2	\IRM.GET.HASHFILE	20		
IRM.CREF.BUTTONEVENTFN	17	IRM.PUT.CREF	17	\IRM.GET.KEYWORDS	20		

VARIABLE INDEX

AROUNDEXITFNS	20	DINFO.GRAPHS	16	IRM.HASHFILE.NAME	3	\IRM.KEYWORDS	20
CLHS.INDEX	5,16	HELPSYS.REPO.TYPES	16	IRM.HOST&DIR	3		
CLHS.OPENER	16	IRM.CREF.FONT	18	\IRM.CREF.IMAGEFNS	18		
CLHS.ROOT.URL	16	IRM.DINFOGRAPH	16	\IRM.HASHFILE	20		

PROPERTY INDEX

HELPSYS	20
---------------	----

COMMAND INDEX

"man"	2
-------------	---

RECORD INDEX

IRMREFERENCE	1
--------------------	---
