File created: 15-Apr-88 09:50:23 **{ERINYES}<LISPUSERS>KOTO>SINGLEFILEINDEX.;2**

changes to: (FNS \SFI.CENTERPRINT)

previous date: 31-Mar-86 17:15:30 {ERINYES}<LISP>KOTO>LISPUSERS>SINGLEFILEINDEX.;1

Read Table: OLD-INTERLISP-FILE

Package: INTERLISP

Format: XCCS


```
(RPAQQ SINGLEFILEINDEXCOMS
       [(COMS (* * "Created by Christopher Tong and JonL White, February 1984.  Heavily revised by Bill van
                  Melle, March 1986.")
              (* SINGLEFILEINDEX)
              (FNS SINGLEFILEINDEX \SFI.Q1UP \FILELISTING SINGLEFILEINDEX2 SINGLEFILEINDEX1 \SFI.AnalyzeLine
                   \SFI.FLUSHFONTCHANGE PrintFnDef INDEXCOPYBYTES INDEXNEWLINE INDEXNEWPAGE \SFI.SORTINDEX
                   UALPHORDERCAR \SFI.FILTER.INDEX)
              (FNS PrintFileTitle \SFI.PRINT.INDEX PrintIndex \SFI.PrintIndexFactors PrintRelativeFunctionIndex
                   \SFI.CENTERPRINT PRINTDOTS \SFI.LISTINGHEADER \SFI.BreakLine))
        (INITVARS (PRINTERDEVICEFILENAME (QUOTE {LPT}))
               (RELATIVEINDEXFLG)
               (SINGLEFILEINDEX.TWOSIDED)
               (SINGLEFILEINDEX.DONTSPAWN)
               (\SFI.PROCESS.COMMANDS)
               (\SFI.PROCESSLOCK (CREATE.MONITORLOCK "SINGLEFILEINDEX"))
               (\SFI.PROCESS)
               (ERRORMESSAGESTREAM T))
        (ADDVARS (SINGLEFILEINDEX.TYPES (MACRO DEFMACRO)
                        (VAR (RPAQ RPAQ? RPAQQ ADDTOVAR)
                             TestForVar T)
                        (VAR READVARS TestForUglyVars)
                        (BITMAP RPAQ TestForBitmap)
                        (CONSTANTS CONSTANTS TestForConstants)
                        (RECORD (eval CLISPRECORDTYPES))
                        (PROPERTY PUTPROPS TestForProp)
                        (COURIERPROGRAM COURIERPROGRAM)
                        (TEMPLATE SETTEMPLATE TestForQuotedType)
                        (I.S.OPR I.S.OPR TestForQuotedType)
                        (RESOURCES PUTDEF TestForResource)
                        (ADVICE READVISE))
               (SINGLEFILEINDEX.PROPERTIES (COPYRIGHT)
                      (READVICE ADVICE))
               (SINGLEFILEINDEX.FILTERS (VAR . CONSTANTS)
                      (VAR . BITMAP)))
        (COMS (* "Functions that find types")
              (FNS TestForType TestForQuotedType TestForVar TestForBitmap TestForProp TestForResource
                   TestForUglyVars TestForGenericDefinition TestForConstants SFI.WHOLE.EXPRESSION
                   SFI.LOOKUP.NAME))
        (DECLARE: EVAL@COMPILE DONTCOPY (MACROS .ERRORSTREAM.)
               (RECORDS SFITYPE)
               (FILES (IMPORT)
                      FILEIO)
               (GLOBALVARS DEFAULTFONT NOTLISTEDFILES)
               (GLOBALVARS FILERDTBL RELATIVEINDEXFLG)
               (GLOBALVARS SINGLEFILEINDEX.DONTSPAWN \SFI.PROCESS.COMMANDS \SFI.PROCESSLOCK \SFI.PROCESS
                      SINGLEFILEINDEX.TWOSIDED SINGLEFILEINDEX.TYPES SINGLEFILEINDEX.PROPERTIES
                      SINGLEFILEINDEX.FILTERS FILELINELENGTH MACROPROPS PRINTERDEVICEFILENAME)
               DONTEVAL@LOAD
               (SPECVARS . T))
        (COMS (FNS SFI.LISTFILES1)
              (DECLARE: DOCOPY DONTEVAL@LOAD (P (MOVD? (QUOTE LISTFILES1)
                                                       (QUOTE OLDLISTFILES1))
                                                (/MOVD (QUOTE SFI.LISTFILES1)
                                                       (QUOTE LISTFILES1)))
                    (INITVARS (LINESPERPAGE 65])
```

(* * "Created by Christopher Tong and JonL White, February 1984. Heavily revised by Bill van Melle, March 1986.")

(* * SINGLEFILEINDEX)

```
(DEFINEQ
```

(SINGLEFILEINDEX

```
  [LAMBDA (INF OUTF mergedIndexFlg PRINTOPTIONS)                    (* bvm: "28-Mar-86 17:31")
    (LET ((FULL (FINDFILE INF T)))
         (COND
            ((NOT FULL)                                             (* When called by LISTFILES INF will already be a full file name)
             (printout (.ERRORSTREAM.)
                    T INF " not found."))
            (SINGLEFILEINDEX.DONTSPAWN (SINGLEFILEINDEX2 FULL OUTF mergedIndexFlg PRINTOPTIONS))
            (T (\SFI.Q1UP (FUNCTION SINGLEFILEINDEX2)
```

```
                FULL OUTF mergedIndexFlg PRINTOPTIONS)      (* Used to return NIL so that LISTFILES won't try removing from
            NOTLISTEDFILES)
            FULL])
```

(\SFI.Q1UP

```
[LAMBDA (FUN FULL OUTF mergedIndexFlg PRINTOPTIONS)          (* bvm: "15-Mar-86 17:11")

        (* * Add a command to list file FULL to OUTF applying FUN)

   (WITH.MONITOR \SFI.PROCESSLOCK                            (* Lock protects \SFI.PROCESS.COMMANDS and
                                                             \SFI.PROCESS)
       [COND
          ((AND \SFI.PROCESS (NOT (FIND.PROCESS \SFI.PROCESS))) (* Process died, flush handle and any old listing requests)
           (SETQ \SFI.PROCESS (SETQ \SFI.PROCESS.COMMANDS NIL]
       (SETQ \SFI.PROCESS.COMMANDS (NCONC1 \SFI.PROCESS.COMMANDS (LIST FUN FULL OUTF mergedIndexFlg
                                                                       PRINTOPTIONS)))
       [COND
          ((NULL \SFI.PROCESS)
           (SETQ \SFI.PROCESS (ADD.PROCESS (LIST (FUNCTION \FILELISTING))
                                           (QUOTE BEFOREEXIT)
                                           (QUOTE DON'T])])
```

(\FILELISTING

```
[LAMBDA NIL                                                  (* bvm: "15-Mar-86 16:58")

        (* * Process that takes listing commands from \SFI.PROCESS.COMMANDS and performs them)

   (WITH.MONITOR \SFI.PROCESSLOCK                            (* Lock protects \SFI.PROCESS.COMMANDS and
                                                             \SFI.PROCESS)
       (while \SFI.PROCESS.COMMANDS bind FORM do (SETQ FORM (pop \SFI.PROCESS.COMMANDS))
                                                 (RELEASE.MONITORLOCK \SFI.PROCESSLOCK)
                                                             (* Release lock while listing so that others can add to my queue)
                                                 (APPLY (CAR FORM)
                                                        (CDR FORM))
                                                 (OBTAIN.MONITORLOCK \SFI.PROCESSLOCK)
         finally                                             (* Nothing left to do, so exit)
               (SETQ \SFI.PROCESS NIL)))])
```

(SINGLEFILEINDEX2

```
[LAMBDA (FULL OUTF mergedIndexFlg PRINTOPTIONS)              (* bvm: "28-Mar-86 17:44")

        (* * Process a single file FULL to OUTF with options. SINGLEFILEINDEX should have already computed the fullname of the
        input file)

   (COND
      ((COND
          ((SINGLEFILEINDEX1 FULL OUTF mergedIndexFlg PRINTOPTIONS)
           (AND (NULL OUTF)
                (printout (.ERRORSTREAM.)
                       T "indexed version of " FULL " => " PRINTERDEVICEFILENAME))
           T)
          (OUTF (printout (.ERRORSTREAM.)
                       T FULL " is not LISPSOURCEFILEP -- COPYFILE being called"))
          (T (OLDLISTFILES1 FULL PRINTOPTIONS)))

        (* Do this here since there is little coordination between the various multiple processes which are listing files)

       (SETQ NOTLISTEDFILES (REMOVE (ROOTFILENAME FULL)
                                    NOTLISTEDFILES))
       NIL])
```

(SINGLEFILEINDEX1

```
[LAMBDA (FULL OUTF RETINDEXFLG PRINTOPTIONS)                 (* bvm: "31-Mar-86 15:53")

        (* Makes an indexed file (default is the line printer)%. The index file will have a number of indices, one for each type in
        INDEXEDTYPESLIST. Each type index will list all the items of that type NIL in alphabetical order and the page number of
        where that item's definition is in the file. -
        NOTE1: The indices will be printed last. -
        NOTE2: The index file is not "loadable" into LISP.)

   (DECLARE (SPECVARS FULL)
            (USEDFREE LINESPERPAGE))
   (RESETLST
       [PROG ((LINESPERPAGE LINESPERPAGE)
              [typesLST (OR (NULL RELATIVEINDEXFLG)
                            (EQ RELATIVEINDEXFLG (QUOTE BOTH]
              (FNUM 0)
              (SOURCESTREAM)
              (PAGECOUNT)
              (LINECOUNT 1)
              (ItemPages)
              (INDICES)
              lastPage MAP FULLEOLC COMS currentItem nextFnGroup nextFnStart FNSMAPSL TEM)
```

```
(DECLARE (SPECVARS MAP LINECOUNT PAGECOUNT LINESPERPAGE SOURCESTREAM ItemPages typesLST FNUM
                   currentItem linePos newPos INDICES))
```

(* * Specials are as follows -
SOURCESTREAM -- stream on the input file being formatted -
currentItem -- function, etc currently being printed -
FNUM -- ordinal number of function currently being printed, when RELATIVEINDEXFLG -
PAGECOUNT -- number of current page -
LINECOUNT -- number of current line on page -
ItemPages -- list of (name type page#) constituting the actual index)

```
[RESETSAVE (SETQ SOURCESTREAM (OPENSTREAM FULL (QUOTE INPUT)
                                       (QUOTE OLD)))
       (QUOTE (PROGN (CLOSEF? OLDVALUE]
(SETQ FULL (FULLNAME SOURCESTREAM))
(COND
   ([EQ FULL (CAR (SETQ TEM (LISTP (GETP (ROOTFILENAME FULL)
                                         (QUOTE FILEMAP]
```

(* It appears as though the file has already been loaded in some way so that the MAP is already loaded)

```
    (SETQ MAP (CADR TEM)))
   ((NULL USEMAPFLG)
    (RESETSAVE NIL (QUOTE (SETQ USEMAPFLG)))
```

(* Really should bind USEMAPFLG to T but this works if the system still thinks it's a globalvar)

```
    (SETQ USEMAPFLG T)))
(COND
   ([OR (AND (NOT (RANDACCESSP SOURCESTREAM))
             (OR typesLST (NULL MAP)))
        (AND (NULL MAP)
             (NULL (SETQ MAP (GETFILEMAP FULL)))
             (NOT (LISPSOURCEFILEP FULL]
```

(* We just let the "old" listfiles do it when the file isn't RANDACCESSP or when it's probably some kind of binary file)

```
    (RETURN)))
(OR OUTF (SETQ OUTF PRINTERDEVICEFILENAME))
[COND
   [(OPENP OUTF (QUOTE OUTPUT))
    (RESETSAVE (OUTPUT (SETQ OUTF (GETSTREAM OUTF (QUOTE OUTPUT]
   (T (RESETSAVE [OUTPUT (SETQ OUTF (OPENSTREAM OUTF (QUOTE OUTPUT)
                                                 (QUOTE NEW]
             (QUOTE (PROGN (CLOSEF? (OUTPUT OLDVALUE]
[STREAMPROP OUTF (QUOTE PRINTOPTIONS)
       (APPEND PRINTOPTIONS (LIST (QUOTE DOCUMENT.NAME)
                                  FULL)
               (STREAMPROP OUTF (QUOTE PRINTOPTIONS]
                                                        (* Make sure printer knows original name of file)
(RESETSAVE (RADIX 10))
(SETQ LINESPERPAGE (OR (GETFILEINFO OUTF (QUOTE PAGEHEIGHT))
                       LINESPERPAGE))                   (* Determine printing parameters.)
(RESETSAVE (LINELENGTH 1000 OUTF))
(COND
   (RELATIVEINDEXFLG                                    (* All index info up front, derived from file map, no absolute page
          numbers)
          (PrintFileTitle FULL (GETFILEINFO SOURCESTREAM (QUOTE CREATIONDATE)))
          (PrintRelativeFunctionIndex MAP)))
[COND
   (typesLST (SETQ typesLST (for ENTRY in SINGLEFILEINDEX.TYPES
                               collect (COND
                                          ((EQ (CAR (LISTP (fetch (SFITYPE PATTERNS) of ENTRY)))
                                               (QUOTE eval))
                                           (create SFITYPE
                                                   PATTERNS _ (EVAL (CADR (fetch (SFITYPE PATTERNS)
                                                                                 of ENTRY)))
                                               reusing ENTRY))
                                          (T ENTRY]
(PROGN (SETQ FNSMAPSL (CDR MAP))
       (SETQ FULLEOLC (fetch EOLCONVENTION of SOURCESTREAM))
       (SETQ PAGECOUNT 1)
       (SETQ nextFnGroup (CDDR (CAR FNSMAPSL)))
       (SETQ nextFnStart (CADAR nextFnGroup)))
```

(* * Locate and print definitions for each item.)

```
(bind linePos newPos (currentPos _ 0)
      [EOL _ (SELECTC FULLEOLC
                  (CR.EOLC (CONCATCODES (CHARCODE (CR))))
                  (LF.EOLC (CONCATCODES (CHARCODE (LF))))
                  (CONCATCODES (CHARCODE (CR LF]
   while (SETQ newPos (FILEPOS EOL SOURCESTREAM currentPos))
   do
```

(* currentPos = how far we have copied; linePos = start of current line;
newPos = start of next line)

```
                    (SETFILEPTR SOURCESTREAM (SETQ linePos currentPos))
                    (COND
                       ([COND
                           [(EQ (PEEKCCODE SOURCESTREAM)
                                (CHARCODE ^F))                    (* Line might start with a fontchange sequence)
                            (\SFI.FLUSHFONTCHANGE SOURCESTREAM) (* Advance linePos to after any font change chars)
                            (AND nextFnStart (OR (IEQP linePos nextFnStart)
                                                 (IEQP currentPos nextFnStart]
                           (T (AND nextFnStart (IEQP linePos nextFnStart]
                                                                  (* Index and print function group.)
                        (for function in nextFnGroup do (SETQ newPos (PrintFnDef function OUTF)))
                                                                  (* Should point us at the first of two closing parens)
                        (pop FNSMAPSL)
                        (SETQ nextFnGroup (CDDAR FNSMAPSL))
                        (SETQ nextFnStart (CADAR nextFnGroup)))
                       (T                                         (* Print and index (when appropriate) next line.)
                          (SELECTC FULLEOLC
                               (CRLF.EOLC (READC SOURCESTREAM)
                                          (add newPos 1))
                               0)
                          (COND
                             (typesLST (\SFI.AnalyzeLine SOURCESTREAM typesLST)))
                          (INDEXCOPYBYTES SOURCESTREAM OUTF currentPos newPos)
                                                                  (* Print the line.)
                          (INDEXNEWLINE)))
                    (SETQ currentPos (ADD1 newPos)))
                 (SETQ lastPage PAGECOUNT)

           (* * Print file index or indices.)

                 (COND
                    ((OR (NULL RELATIVEINDEXFLG)
                         (EQ RELATIVEINDEXFLG (QUOTE BOTH)))
                     (SETQ INDICES (\SFI.SORTINDEX ItemPages))
                     [LET ((VARS (ASSOC (QUOTE VAR)
                                        INDICES)))                (* Manually filter out the filecoms var)
                          (RPLACD VARS (DREMOVE (ASSOC (FILECOMS FULL)
                                                       (CDR VARS))
                                                (CDR VARS]
                     (\SFI.FILTER.INDEX INDICES)
                     (INDEXNEWPAGE T)
                     (COND
                        ((AND (EVENP PAGECOUNT)
                              SINGLEFILEINDEX.TWOSIDED)           (* Ensure that the index will not be on the back-side of a
                                                                  two-sided listing)
                         (INDEXNEWPAGE T)))
                     (PrintFileTitle FULL (GETFILEINFO SOURCESTREAM (QUOTE CREATIONDATE)))
                     (\SFI.PRINT.INDEX INDICES)))
                 (RETURN (COND
                            (RETINDEXFLG (CONS FULL INDICES))
                            (T FULL])])
```

(\SFI.AnalyzeLine

```
  [LAMBDA (SOURCESTREAM TYPETRIPLES FLG)                          (* bvm: "30-Mar-86 15:07")
```

(* * Retrieve line as string, beginning with first character that isn't a font change char,)

```
    (DECLARE (USEDFREE ItemPages))
    (SELECTQ (GETSYNTAX (READCCODE SOURCESTREAM)
                    FILERDTBL)
         ((LEFTPAREN LEFTBRACKET)
```

(* Note that if the first character on the line isn't a parens then this line can't be the start of anything interesting)

```
            (COND
               ((EQ (PEEKCCODE SOURCESTREAM)
                    (CHARCODE ^F))
```

(* It is possible to have a fontchange sequence just after the open parens, though most forms reserve the font change for the named object, coming up next)

```
                (\SFI.FLUSHFONTCHANGE SOURCESTREAM)))
            (LET ((FN (READ SOURCESTREAM FILERDTBL))
                  HERE PAT MOVED? ITEMNAME)
                 (SETQ HERE (GETFILEPTR SOURCESTREAM))
                 (for ENTRY in TYPETRIPLES when (COND
                                                   ((EQ (SETQ PAT (fetch (SFITYPE PATTERNS) of ENTRY))
                                                        T)        (* Matches anything -- TESTFN must be doing all the work)
                                                    T)
                                                   ((LISTP PAT)
                                                    (MEMB FN PAT))
                                                   (T (EQ FN PAT)))
                    do                                            (* ENTRY thinks this line might be interesting)
                        (COND
                           (MOVED?                                (* Previous test may have moved the file pointer, so bring it
                                                                  back)
```

```
                                (SETFILEPTR SOURCESTREAM HERE)
                                (SETQ MOVED? NIL)))
                          [COND
                             ([SETQ ITEMNAME (CAR (NLSETQ (APPLY* (OR (fetch (SFITYPE TESTFN) of ENTRY)
                                                                      (FUNCTION TestForType))
                                                                  SOURCESTREAM FN ENTRY]
                              [COND
                                 ((NLISTP ITEMNAME)                (* Single object to be indexed as the type in ENTRY)
                                  (push ItemPages (LIST (LET ((TYPE (fetch (SFITYPE NAME) of ENTRY)))
                                                          (OR (CAR (LISTP TYPE))
                                                              TYPE))
                                                        ITEMNAME PAGECOUNT)))
                                 (T                                (* Index as some other type)
                                    (for PAIR in (COND
                                                    ((LITATOM (CAR ITEMNAME))
                                                                   (* a single pair)
                                                     (LIST ITEMNAME))
                                                    (T             (* many)
                                                       ITEMNAME))
                                       do (for NAME in (CDR PAIR) do (push ItemPages (LIST (CAR PAIR)
                                                                                           NAME PAGECOUNT]
                              (COND
                                 ((NOT (fetch (SFITYPE AMBIGUOUS?) of ENTRY))
                                  (RETURN]
                          (SETQ MOVED? T))))
       ((RIGHTPAREN RIGHTBRACKET)

         (* Well, some lines will be the closing of a DEFINEQ or a DECLARE: or whatever)

          NIL)
       NIL])
```

(\SFI.FLUSHFONTCHANGE

```
[LAMBDA (STREAM)                                                (* bvm: "15-Mar-86 17:41")
   (while (EQ (PEEKCCODE STREAM)
              (CHARCODE ^F))
      do (READCCODE STREAM)
         (READCCODE STREAM)
         (add linePos 2])
```

(PrintFnDef

```
[LAMBDA (FNDEF OUTSTREAM)
   (DECLARE (USEDFREE ItemPages FNUM SOURCESTREAM PAGECOUNT LINESPERPAGE LINECOUNT)
            (SPECVARS currentItem))                               (* bvm: "28-Mar-86 17:41")

         (* * Prints a FNDEF definition on the file OUTSTREAM -
         FNDEF is map entry of form (name start . end))

   (PROG ((END (CDDR FNDEF))
          (currentItem (CAR FNDEF)))
         (add FNUM 1)
         (INDEXNEWLINE)
         (COND
            (RELATIVEINDEXFLG (printout NIL .SP (IDIFFERENCE FILELINELENGTH (IPLUS 2 (NCHARS FNUM)))
                                        .FONT BOLDFONT "[" FNUM "]" .FONT DEFAULTFONT .RESET)))
         (INDEXNEWLINE)
         (COND
            ((NOT (ILEQ (IPLUS LINECOUNT 3)
                        LINESPERPAGE))
             (INDEXNEWPAGE)))
         (push ItemPages (LIST (QUOTE FUNCTION)
                               currentItem PAGECOUNT))            (* Print out function.)
         (INDEXCOPYBYTES SOURCESTREAM OUTSTREAM (CADR FNDEF)
                END)
         (RETURN END])
```

(INDEXCOPYBYTES

```
[LAMBDA (IN OUT START END)
   (DECLARE (USEDFREE LINECOUNT LINESPERPAGE))                    (* bvm: "15-Mar-86 17:50")

         (* This is similar to COPYBYTES except that, INDEXNEWLINE is called whenever an EOL is read, and IndexNewPage is
         called whenever a form feed is read)

   (SETFILEPTR IN START)
   [PROG ((INSTRM (GETSTREAM IN (QUOTE INPUT)))
          (OUTSTRM (GETSTREAM OUT (QUOTE OUTPUT)))
          EOLC NLFLG LOOKFORLF CH)
         (SETQ EOLC (fetch EOLCONVENTION of INSTRM))
         (FRPTQ (IDIFFERENCE END START)
                (SELCHARQ (SETQ CH (BIN INSTRM))
                     (CR [SELECTC EOLC
                             (CR.EOLC (SETQ LOOKFORLF NIL)
                                      (COND
                                         ((AND NLFLG (IGREATERP LINECOUNT (IDIFFERENCE LINESPERPAGE 5)))
```

```
                                                                    (* double cr near end of page)
                                        (INDEXNEWPAGE)
                                        (SETQ NLFLG NIL))
                                       (T (INDEXNEWLINE)
                                          (SETQ NLFLG T))))
                              (CRLF.EOLC
```

(* Flag says that EOLC is CRLF and we are looking for next char to be LF.
Expanded out this way so that we can keep track of the character counts accurately)

```
                                          (SETQ LOOKFORLF T))
                                    (PROGN (SETQ LOOKFORLF NIL)
                                           (\OUTCHAR OUTSTRM (CHARCODE CR])
                             (LF [COND
                                    [(OR LOOKFORLF (EQ EOLC LF.EOLC))
                                     (COND
                                        ((AND NLFLG (IGREATERP LINECOUNT (IDIFFERENCE LINESPERPAGE 5)))
                                                                    (* double cr near end of page)
                                         (INDEXNEWPAGE)
                                         (SETQ NLFLG NIL))
                                        (T (INDEXNEWLINE)
                                           (SETQ NLFLG T]
                                    (T (\OUTCHAR OUTSTRM (CHARCODE LF))
```

(* If LF comes thru, it is just a vertical tab. Want to keep horizontal position the same, but update line-counts)

```
                                   (COND
                                      ((AND NLFLG (IGREATERP LINECOUNT (IDIFFERENCE LINESPERPAGE 5)))
                                                                    (* double cr near end of page)
                                       (INDEXNEWPAGE)
                                       (SETQ NLFLG NIL))
                                      (T (COND
                                            ((IGREATERP (add LINECOUNT 1)
                                                    LINESPERPAGE)
                                             (INDEXNEWPAGE)))
                                         (SETQ NLFLG T]
                                (SETQ LOOKFORLF NIL))
                             (FF (INDEXNEWPAGE)
                                 (SETQ NLFLG NIL)
                                 (SETQ LOOKFORLF NIL))
                             (PROGN (\BOUT OUTSTRM CH)
                                    (SETQ NLFLG NIL)
                                    (SETQ LOOKFORLF NIL]
    T])
```

(INDEXNEWLINE

```
  [LAMBDA (DontPrintPageNbrFlg)                                     (* JonL "13-Mar-84 22:04")
    (TERPRI)
    (COND
       ((IGREATERP (add LINECOUNT 1)
                LINESPERPAGE)
        (INDEXNEWPAGE DontPrintPageNbrFlg])
```

(INDEXNEWPAGE

```
  [LAMBDA (DontPrintPageNbrFlg)                                     (* JonL "13-Mar-84 22:04")
    (PRIN3 (FCHARACTER (CHARCODE FF)))
    (POSITION NIL 0)
    (SETQ LINECOUNT 0)
    (COND
       (PAGECOUNT (add PAGECOUNT 1)))
    (\SFI.LISTINGHEADER DontPrintPageNbrFlg])
```

(\SFI.SORTINDEX

```
  [LAMBDA (TRIPLES)                                                 (* bvm: "29-Mar-86 17:26")
```

(* * Sort TRIPLES into a set of indices, one per type. Each element is of the form
(type name page), while the resulting indices are of the form (type . entries), with each entry looking like
(name . pagenumbers))

```
    (LET ([TYPENAMES (CONS (QUOTE FUNCTION)
                           (for X in SINGLEFILEINDEX.TYPES collect (CAR X]
          RESULT INDEX OLDNAME)
         [for TRIP in TRIPLES do [COND
                                    ((NULL (SETQ INDEX (ASSOC (CAR TRIP)
                                                              RESULT)))
                                     (push RESULT (SETQ INDEX (LIST (CAR TRIP]
                                   (COND
                                      [(SETQ OLDNAME (ASSOC (CADR TRIP)
                                                            INDEX)) (* Duplicate entry, so add a page number)
                                       (RPLACD OLDNAME (SORT (UNION (CDDR TRIP)
                                                                    (CDR OLDNAME]
                                      (T (push (CDR INDEX)
                                               (CDR TRIP]
         (for PAIR in RESULT do (SORT (CDR PAIR)
```

```
                                    (FUNCTION UALPHORDERCAR)))
          (SORT RESULT (FUNCTION (LAMBDA (X Y)                    (* X is before Y if its car appears before Y's in TYPENAMES)
                                   (FMEMB (CAR Y)
                                          (CDR (FMEMB (CAR X)
                                                      TYPENAMES])
```

(UALPHORDERCAR

```
  [LAMBDA (A B)                                                   (* JonL " 7-Mar-84 19:52")
                                                                  (* does case independent sort on the CAR of two elements.)
    (UALPHORDER (CAR A)
           (CAR B])
```

(\SFI.FILTER.INDEX

```
  [LAMBDA (INDICES)                                               (* bvm: "30-Mar-86 14:11")

          (* * Remove redundancies from the prepared INDICES)

    (DECLARE (SPECVARS INDICES))                                  (* For SFI.LOOKUP.NAME)
    (for TYPEPAIR in INDICES bind FILTERS when [SETQ FILTERS (for FILTER in SINGLEFILEINDEX.FILTERS
                                                                collect (CDR FILTER)
                                                                when (EQ (CAR FILTER)
                                                                         (CAR TYPEPAIR]
       do
          (* Each filter is either a type name or a list whose car is a function)

          (RPLACD TYPEPAIR (for PAIR in (CDR TYPEPAIR) collect PAIR
                              unless (for F in FILTERS thereis (COND
                                                                 ((NLISTP F)
                                                                   (* Name exists as another type)
                                                                  (SFI.LOOKUP.NAME (CAR PAIR)
                                                                         F))
                                                                 (T (APPLY* (CAR F)
                                                                            PAIR])

)

(DEFINEQ
```

(PrintFileTitle

```
  [LAMBDA (FILENAME DATE)                                         (* bvm: "15-Mar-86 17:17")

          (* * Print FILENAME title. Should not be called unless FILENAME is essentially "at the top of the page")

    (\SFI.CENTERPRINT (CONCAT FILENAME "          " DATE)
          T)
    (\SFI.CENTERPRINT (CONCAT "-- Listed on " (DATE)
                              " --"))
    (INDEXNEWLINE])
```

(\SFI.PRINT.INDEX

```
  [LAMBDA (INDICES)                                               (* bvm: "30-Mar-86 15:52")

          (* * For each (type . entries) pair in INDICES print a pretty index for the items of the type)

    (for PAIR in INDICES when (CDR PAIR) do (PrintIndex (CDR PAIR)
                                                        lastPage
                                                        (CAR PAIR))
                                            (INDEXNEWLINE T))
    (\SFI.BreakLine])
```

(PrintIndex

```
  [LAMBDA (INDEXPAIRS MaxIndexNo TYPE)                            (* bvm: "30-Mar-86 15:34")

          (* * print index of items in IndexedList.)

    (DECLARE (USEDFREE LINESPERPAGE LINECOUNT))
    (PROG ([INDEXNOWIDTH (COND
                           ((ILESSP MaxIndexNo 10)
                            1)
                           ((ILESSP MaxIndexNo 100)
                            2)
                           (T (NCHARS MaxIndexNo]
           NCOLUMNS NROWS WIDTH LEFT SPACING NROWSREMAINING LastItem)
          (DECLARE (SPECVARS NCOLUMNS LEFT WIDTH SPACING NROWS))
          (SETQ WIDTH (IPLUS (for PAIR in INDEXPAIRS bind largest (PLUS (NCHARS (CAR PAIR))
                                                                        (COND
                                                                           ((CDDR PAIR)

          (* When multiple page nos, must count the extra pages, plus an additional char each for the separating comma)

                                                                             (ITIMES (LENGTH (CDDR PAIR))
                                                                                    (IPLUS 1 INDEXNOWIDTH)))
                                                                           (T 0)))
```

```
                        finally (RETURN $$EXTREME))
                     INDEXNOWIDTH 1))                    (* WIDTH is the widest any entry gets: name plus page numbers)
(\SFI.PrintIndexFactors INDEXPAIRS)                      (* Compute NCOLUMNS LEFT WIDTH SPACING NROWS)
(SETQ NROWSREMAINING NROWS)
(AND TYPE (\SFI.BreakLine))                              (* When TYPE is non-null, call is from PrintOneTypeIndex)
(INDEXNEWLINE T)
(COND
   (TYPE [COND
            ((AND (IGREATERP (IPLUS NROWS 3)
                             (IDIFFERENCE LINESPERPAGE LINECOUNT))
                  (IGREATERP LINECOUNT (LRSH LINESPERPAGE 1)))
```

(* * Don't start an indexing on the bottom half of a page which is going to cross a page boundary before the "breaker")

```
             (INDEXNEWPAGE T)
             (AND TYPE (\SFI.BreakLine]
          (\SFI.CENTERPRINT (CONCAT TYPE " INDEX")
                 T T)
          (INDEXNEWLINE T)))
(while INDEXPAIRS
   do (SETQ NROWS (IMIN NROWSREMAINING (IDIFFERENCE LINESPERPAGE LINECOUNT)))
      (for ROW from 1 to NROWS bind NEXTINDEX
         do (SETQ NEXTINDEX ROW)
            (for COLUMN from 1 to NCOLUMNS
               do [COND
                     ((SETQ LastItem (FNTH INDEXPAIRS NEXTINDEX))
                      (LET* ((ITEM (CAR LastItem))
                             (LABEL (CAR ITEM))
                             (PAGENO (CDR ITEM)))
                            [SETQ PAGENO (COND
                                            [(LISTP PAGENO)
                                                        (* More than one occurrence)
                                             (CONCATLIST (CDR (for P in PAGENO
                                                                 join (LIST "," P]
                                            (T (MKSTRING PAGENO]
                            (printout NIL .FONT DEFAULTFONT LABEL ,)
                            (PRINTDOTS (IDIFFERENCE (IDIFFERENCE WIDTH (ADD1 (NCHARS LABEL)))
                                                    (NCHARS PAGENO)))
                            (PRIN1 PAGENO)
                            (COND
                               ((NEQ COLUMN NCOLUMNS)
                                (SPACES SPACING]
                  (add NEXTINDEX NROWS))
            (INDEXNEWLINE T))
      (COND
         ((SETQ INDEXPAIRS (CDR LastItem))
          (INDEXNEWPAGE T)
          (SETQ NROWSREMAINING (ADD1 (IQUOTIENT (LENGTH INDEXPAIRS)
                                                NCOLUMNS])
```

(\SFI.PrintIndexFactors

```
  [LAMBDA (IndexedList)                                  (* bvm: "30-Mar-86 15:00")
    (DECLARE (USEDFREE NCOLUMNS LEFT WIDTH SPACING NROWS))
    (LET ((LEN (LENGTH IndexedList)))
         [SETQ NCOLUMNS (IMAX 1 (IMIN LEN (IQUOTIENT FILELINELENGTH (IPLUS WIDTH 2]
                                                        (* Number of columns that fit if you allow 2 spaces between
                                                        columns)
         (SETQ NROWS (IQUOTIENT (IPLUS LEN (SUB1 NCOLUMNS))
                                NCOLUMNS))
         (SETQ NCOLUMNS (IQUOTIENT (IPLUS LEN (SUB1 NROWS))
                                   NROWS))
```

(* This might reduce the number of columns if all the items, printed in NROWS rows, take fewer columns than originally allocated)

```
         (SETQ LEFT (IDIFFERENCE FILELINELENGTH (ITIMES (IPLUS WIDTH 2)
                                                        NCOLUMNS)))
                                                        (* LEFT is number of spaces remaining after allocating the
                                                        columns)
         (COND
            ((EQ NCOLUMNS 1)
```

(* Only one column, so either make it half the page width or the full width)

```
             [SETQ WIDTH (COND
                            ((GREATERP WIDTH (IQUOTIENT FILELINELENGTH 2))
                             FILELINELENGTH)
                            (T (IQUOTIENT FILELINELENGTH 2]
             (SETQ SPACING 0))
            (T (SETQ WIDTH (IMIN (IPLUS WIDTH (IQUOTIENT LEFT 2))
                                 (IDIFFERENCE (IQUOTIENT FILELINELENGTH NCOLUMNS)
                                              2)))       (* Spaces LEFT gets divided between the dots an the
                                                         between-column spaces.)
               (SETQ SPACING (COND
                                ((EQ NCOLUMNS 1)
                                 0)
```

```
                                (T (IQUOTIENT (IDIFFERENCE FILELINELENGTH (ITIMES WIDTH NCOLUMNS))
                                              (SUB1 NCOLUMNS])
```

(PrintRelativeFunctionIndex

```
  [LAMBDA (MAP)                                                    (* bvm: "31-Mar-86 15:59")

          (* * Create and print an index for the functions on the file.)

    (PROG ((MaxIndexNo 0)
           IndexedList currentItem)
          [SETQ IndexedList (for DFQ in MAP join (for function in (CDDR DFQ) collect (LIST (CAR function)
                                                                                            (add MaxIndexNo 1]
                                                                   (* Printout function index.)
          (COND
             ((NOT IndexedList)
              (INDEXNEWLINE T)
              (INDEXNEWLINE T)
              (printout NIL .FONT BOLDFONT "No Functions." .FONT DEFAULTFONT))
             (T (PrintIndex (SORT IndexedList (FUNCTION UALPHORDERCAR))
                            MaxIndexNo)))
          (INDEXNEWPAGE T)
          (RETURN MAP])
```

(\SFI.CENTERPRINT

```
  [LAMBDA (STR BOLDFLG DontPrintPageNbrFlg)                        (* //Z\\ "15-Apr-88 09:49")
                                                                   (* JonL "13-Mar-84 22:07")
                                                                   (* Be sure to only TAB with a positive index)
    (TAB (IQUOTIENT (if (IGREATERP FILELINELENGTH (NCHARS STR))
                        then (IDIFFERENCE FILELINELENGTH (NCHARS STR))
                      else 0)
                    2))
    (COND
       (BOLDFLG (printout NIL .FONT BOLDFONT STR .FONT DEFAULTFONT))
       (T (printout NIL STR)))
    (INDEXNEWLINE DontPrintPageNbrFlg])
```

(PRINTDOTS

```
  [LAMBDA (N FILE)                                                 (* bvm: "15-Mar-86 16:28")
    (LET [(STRM (GETSTREAM FILE (QUOTE OUTPUT]
         (FRPTQ N (\OUTCHAR STRM (CHARCODE %.])
```

(\SFI.LISTINGHEADER

```
  [LAMBDA (dontPrintPageNumberFlg)                                 (* cht: " 5-JAN-84 15:15")
    (COND
       (FULL (PRIN1 FULL)))
    (COND
       ((AND currentItem FNUM RELATIVEINDEXFLG)
        (printout NIL "  (" .P2 currentItem "[" FNUM "] cont.)"))
       (currentItem (printout NIL "  (" .P2 currentItem " cont.)")))
    (TAB (IDIFFERENCE FILELINELENGTH 9)
         T)
    (COND
       ((AND PAGECOUNT (NOT dontPrintPageNumberFlg))
        (PRIN1 "Page ")
        (PRINTNUM (QUOTE (FIX 4))
                  PAGECOUNT)))
    (INDEXNEWLINE)
    (INDEXNEWLINE])
```

(\SFI.BreakLine

```
  [LAMBDA NIL                                                      (* bvm: "15-Mar-86 16:28")
    (INDEXNEWLINE T)
    [LET [(STRM (GETSTREAM NIL (QUOTE OUTPUT]
         (FRPTQ FILELINELENGTH (\OUTCHAR STRM (CHARCODE ~]
    (INDEXNEWLINE T])

)

(RPAQ? PRINTERDEVICEFILENAME (QUOTE {LPT}))

(RPAQ? RELATIVEINDEXFLG )

(RPAQ? SINGLEFILEINDEX.TWOSIDED )

(RPAQ? SINGLEFILEINDEX.DONTSPAWN )

(RPAQ? \SFI.PROCESS.COMMANDS )

(RPAQ? \SFI.PROCESSLOCK (CREATE.MONITORLOCK "SINGLEFILEINDEX"))

(RPAQ? \SFI.PROCESS )
```

```
(RPAQ? ERRORMESSAGESTREAM T)

(ADDTOVAR SINGLEFILEINDEX.TYPES
          (MACRO DEFMACRO)
          (VAR (RPAQ RPAQ? RPAQQ ADDTOVAR)
               TestForVar T)
          (VAR READVARS TestForUglyVars)
          (BITMAP RPAQ TestForBitmap)
          (CONSTANTS CONSTANTS TestForConstants)
          (RECORD (eval CLISPRECORDTYPES))
          (PROPERTY PUTPROPS TestForProp)
          (COURIERPROGRAM COURIERPROGRAM)
          (TEMPLATE SETTEMPLATE TestForQuotedType)
          (I.S.OPR I.S.OPR TestForQuotedType)
          (RESOURCES PUTDEF TestForResource)
          (ADVICE READVISE))

(ADDTOVAR SINGLEFILEINDEX.PROPERTIES (COPYRIGHT)
                                     (READVICE ADVICE))

(ADDTOVAR SINGLEFILEINDEX.FILTERS (VAR . CONSTANTS)
                                  (VAR . BITMAP))
```

;; Functions that find types

```
(DEFINEQ
```

(TestForType

```
  [LAMBDA (STREAM FN TRIPLE)                                  (* bvm: "30-Mar-86 13:20")
```

(* * Default testfn for types that are dumped in a form whose second element is the object's name)

```
    (LET ((NAME (READ STREAM FILERDTBL)))
         (AND NAME (LITATOM NAME)
              NAME])
```

(TestForQuotedType

```
  [LAMBDA (STREAM FN TRIPLE)                                  (* bvm: "30-Mar-86 13:29")
```

(* * Like TestForType, but tests for something where the second element of the form is the quoted name.)

```
    (LET ((NAME (READ STREAM FILERDTBL)))
         (AND (EQ (CAR (LISTP NAME))
                  (QUOTE QUOTE))
              (CADR NAME])
```

(TestForVar

```
  [LAMBDA (STREAM FN TRIPLE)                                  (* bvm: "29-Mar-86 17:02")
```

(* * Called for expressions whose car is one of RPAQ, RPAQQ, RPAQ?, ADDTOVAR --
read the variable name following it. Filters after the fact will remove duplications with other variable types)

```
    (LET (NAME)
         (COND
            ([AND (SETQ NAME (READ STREAM FILERDTBL))
                  (LITATOM NAME)
                  (NEQ NAME T)
                  (NOT (FMEMB NAME (QUOTE (GLOBALVARS SPECVARS LOCALVARS NLAMA NLAML LAMA]
                                                              (* Ignore compiler-internal vars)
             NAME])
```

(TestForBitmap

```
  [LAMBDA (STREAM FN TRIPLE)                                  (* bvm: "28-Mar-86 17:06")
```

(* * Called on (RPAQ --) in case the expression is (RPAQ var (READBITMAP)))

```
    (LET ((NAME (READ STREAM FILERDTBL))
          CHAR)
         (COND
            ([AND NAME (LITATOM NAME)
                  (EQ (SETQ CHAR (SKIPSEPRCODES STREAM FILERDTBL))
                      (CHARCODE "("))
                  (PROGN (READCCODE STREAM)                   (* After the VARS name is the form
                                                              (READBITMAP ...))
                         (EQ (RATOM STREAM FILERDTBL)
                             (QUOTE READBITMAP]
             NAME])
```

(TestForProp

```
  [LAMBDA (STREAM FN TRIPLE)                                  (* bvm: "31-Mar-86 12:13")
```

(* * Called when given a PUTPROPS expression. Determine what type it is by looking at the property name.
If no more specific type known, then index it as a PROPERTY)

```
    (LET ((NAME (READ STREAM FILERDTBL))
          (PROP (READ STREAM FILERDTBL)))                    (* See if PROP means something more specific than "property")
         (COND
            ((MEMB PROP MACROPROPS)                          (* Do macros in line so that MACRONAMES can be dynamically
             changed.)
             (LIST (QUOTE MACRO)
                   NAME))
            (T (for PAIR in SINGLEFILEINDEX.PROPERTIES when (EQ (CAR PAIR)
                                                                PROP)
                  do                                         (* Index it under this other type)
                     (RETURN (AND (CADR PAIR)
                                  (LIST (CADR PAIR)
                                        NAME)))
                  finally                                    (* Nothing better, so index it as having a property)
                        (RETURN NAME])
```

(TestForResource

```
  [LAMBDA (STREAM FN TRIPLE)                                 (* bvm: "28-Mar-86 17:08")
    (TestForGenericDefinition STREAM FN (QUOTE ((RESOURCES GLOBALRESOURCES])
```

(TestForUglyVars

```
  [LAMBDA (STREAM FN TRIPLE)                                 (* bvm: "30-Mar-86 15:42")

          (* * Uglyvars are dumped as (READVARS var1 var2 ...))

    (CONS (QUOTE VAR)
          (CDR (SFI.WHOLE.EXPRESSION STREAM])
```

(TestForGenericDefinition

```
  [LAMBDA (STREAM FN TRIPLE)                                 (* bvm: "31-Mar-86 12:02")

          (* * Tests to see if expression is of the form (PUTDEF (QUOTE name)
          (QUOTE type) (QUOTE value)) where type is one specified in TRIPLE)

    (LET ((DESIREDTYPE (CAR TRIPLE))
          NAME TYPE)
         (COND
            ([AND (PROGN                                     (* After the PUTDEF should find (QUOTE name))
                          (EQ [CAR (LISTP (SETQ NAME (READ STREAM FILERDTBL]
                              (QUOTE QUOTE)))
                  (PROGN                                     (* then (QUOTE DESIREDTYPE))
                          (EQ [CAR (LISTP (SETQ TYPE (READ STREAM FILERDTBL]
                              (QUOTE QUOTE)))
                  (OR (EQ [SETQ TYPE (CAR (LISTP (CDR TYPE]
                          DESIREDTYPE)
                      (AND (LISTP DESIREDTYPE)
                           (MEMB TYPE DESIREDTYPE]
             (CADR NAME])
```

(TestForConstants

```
  [LAMBDA (STREAM FN TRIPLE)                                 (* bvm: "30-Mar-86 14:17")

          (* * Called when expression is (CONSTANTS --) -- return all elements
          (or CAR of element when it's a pair) as type CONSTANTS)

    (CONS (QUOTE CONSTANTS)
          (for X in (CDR (SFI.WHOLE.EXPRESSION STREAM)) collect (COND
                                                                ((LISTP X)
                                                                 (CAR X))
                                                                (T X])
```

(SFI.WHOLE.EXPRESSION

```
  [LAMBDA (STREAM)                                           (* bvm: "30-Mar-86 13:34")
    (DECLARE (USEDFREE linePos))

          (* * Called by testfns that want to see the whole expression)

    (SETFILEPTR STREAM linePos)
    (READ STREAM FILERDTBL])
```

(SFI.LOOKUP.NAME

```
  [LAMBDA (NAME TYPE)                                        (* bvm: "30-Mar-86 13:44")
    (ASSOC NAME (CDR (ASSOC TYPE INDICES])

)

(DECLARE: EVAL@COMPILE DONTCOPY

(DECLARE: EVAL@COMPILE
```

```
(PUTPROPS .ERRORSTREAM. MACRO (NIL (SELECTQ ERRORMESSAGESTREAM
                                           (T PROMPTWINDOW)
                                           ERRORMESSAGESTREAM)))
)

(DECLARE: EVAL@COMPILE

(RECORD SFITYPE (NAME PATTERNS TESTFN AMBIGUOUS?))
)

(FILESLOAD (IMPORT)
       FILEIO)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DEFAULTFONT NOTLISTEDFILES)
)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS FILERDTBL RELATIVEINDEXFLG)
)

(DECLARE: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SINGLEFILEINDEX.DONTSPAWN \SFI.PROCESS.COMMANDS \SFI.PROCESSLOCK \SFI.PROCESS
       SINGLEFILEINDEX.TWOSIDED SINGLEFILEINDEX.TYPES SINGLEFILEINDEX.PROPERTIES SINGLEFILEINDEX.FILTERS
       FILELINELENGTH MACROPROPS PRINTERDEVICEFILENAME)
)


(DECLARE: DOEVAL@COMPILE DONTCOPY

(SPECVARS . T)
)
)

(DEFINEQ

(SFI.LISTFILES1
  [LAMBDA (FILE PRINTOPTIONS)                                    (* rmk: "26-Feb-85 10:36")
    (SINGLEFILEINDEX FILE NIL NIL PRINTOPTIONS])

)

(DECLARE: DOCOPY DONTEVAL@LOAD

(MOVD? (QUOTE LISTFILES1)
       (QUOTE OLDLISTFILES1))

(/MOVD (QUOTE SFI.LISTFILES1)
       (QUOTE LISTFILES1))

(RPAQ? LINESPERPAGE 65)
)

(PUTPROPS SINGLEFILEINDEX COPYRIGHT ("Xerox Corporation" 1984 1985 1986 1988))
```

FUNCTION INDEX

VARIABLE INDEX

RECORD INDEX

MACRO INDEX