```
;; Ca2.lsp dictionary lookup routines  12/1/79 W. Lewis Johnson
;All except build-lex-info taken from Larry Birnbaum's Ca2.mll
;which was originally cannibalized from EDL
;

;ASSOCFILE reads S-expressions from the files in FILELIST until one is found
;whose CAR is X; ie., a file is like one old association list, but without
;parentheses at the outer level;

(DE ASSOCFILE (X FILELIST)
  (PROG (:PPN :DEV :FILE :PREV OLDI OLDO ENTRY)
        (SETQ :PPN #(0. 0.))
        (SETQ :DEV #DSK:)
        (SETQ FILELIST (NEXTFILE FILELIST NIL))
        (SETQ OLDI :PREV)
 CHECKF (COND ((INCONTENTS X
                           (COND ((ATOM :FILE) :FILE) (T (CAR :FILE))))
               (GO NEXTE)))
  NEXTF (COND ((SETQ FILELIST (NEXTFILE FILELIST T))
               (GO CHECKF))
              (T (INC OLDI T) (RETURN NIL))))
  NEXTE (COND ((ATOM (ERRSET (SETQ ENTRY (READ)) NIL)) (GO NEXTF)))
        (COND ((AND (NOT (ATOM ENTRY)) (EQUAL X (CAR ENTRY)))
               (INC OLDI T)
               (RETURN ENTRY))
              (T (GO NEXTE)))))


;NEXTFILE takes a DSKIN format list like (FOO: BAZ (22 12) A B SAM: (D.LSP)),
;uses FILESCAN to set the current :PPN, :DEVice and :FILE, opens :FILE for
;input, sets :PREV to the previous channel, closes :PREV if CLOSE is non-NIL,
;and returns the rest of the file list;
(DE NEXTFILE (FILES CLOSE)
  (PROG (TEMP)
        (COND ((NULL FILES) (RETURN NIL)))
        (SETQ TEMP (FILESCAN FILES :DEV :PPN))
        (SETQ FILES (CDR TEMP))
        (SETQ :DEV (CAAR TEMP))
        (SETQ :PPN (CADAR TEMP))
        (COND ((ATOM (SETQ :FILE (CADDAR TEMP)))
               (SETQ :FILE (CONS :FILE #IDX))))
        (SETQ :PREV
              (EVAL (LIST #INC
                          (LIST #INPUT (GENSYM) :DEV :PPN :FILE)
                          CLOSE)))
        (RETURN FILES)))


;FILESCAN scans down the file-list looking for the first file-name, resetting
;the device and PPN along the way as necessary;
(DE FILESCAN (FILE-LIST DEVICE PPN)
  (PROG (TEMP)
   LOOP (COND ((NULL FILE-LIST)
               (RETURN (CONS (LIST DEVICE PPN NIL) NIL))))
        (SETQ TEMP (CAR FILE-LIST))
        (SETQ FILE-LIST (CDR FILE-LIST))
        (COND ((ATOM TEMP)
               (COND ((EQUAL (CAR (LAST (EXPLODE TEMP))) #:)
                      (SETQ DEVICE TEMP)
                      (GO LOOP))
```

```
                (IF (RETURN (CONS (LIST DEVICE PPN TEMP) FILE-LIST))
                 ))
               ((AND (NUMBERP (CAR TEMP)) (NUMBERP (CADR TEMP)))
                (SETQ PPN TEMP)
                (GO LOOP))
               (IF (RETURN (CONS (LIST DEVICE PPN TEMP) FILE-LIST)))))


;INCONTENTS checks the associated content list, if any, for the entry;
(DE INCONTENTS (X FILE)
  (OR (NULL (GET FILE #CONTENTS)) (MEMBER X (GET FILE #CONTENTS))))


;Get-lex-into builds a lexical entry and sticks a request pool in it
(de get-lex-into (dest word)
 (let (new (new-lex))
  (cond  ((null (get word 'requests:))
          (look-up word)))
  (putprop new (build-pool (get word 'requests:)) 'pool:)
  (set dest new)))

;LOOK-UP finds the index entry and then searches
;the correct dictionary for word.
(DE LOOK-UP (WORD)
  (PROG (ENTRY IC)
        (COND ((NULL (SETQ ENTRY (ASSOCFILE WORD :INDICES)))
               (RETURN NIL)))
        (SETQ IC
              (EVAL (LIST #INC
                          (LIST #INPUT
                                (GENSYM)
                                (CADR ENTRY)
                                (CADDR ENTRY))
                          NIL)))
   LOOP (COND ((NULL (NEXTENTRY)) (RETURN NIL))
;          If it got here the entry was found;
              ((EQ WORD (READ)) (GO LOOP))
              (IF (EVAL (READ)) (INC IC T) (RETURN T)))))
```