File created: 9-Mar-89 15:24:58 **{ERINYES}<LISPUSERS>MEDLEY>ANALYZER.;9**

changes to: (FNS Analyzer.ReadWordList)

previous date: 13-Jan-89 15:50:22 {ERINYES}<LISPUSERS>MEDLEY>ANALYZER.;8

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ ANALYZERCOMS
       [(COMS

;;; THE ANALYZER CLASS

              (RECORDS Morphalyzer)
              ;; renamed record to avoid a conflict.
              (MACROS Analyzer.Open Analyzer.Close Analyzer.Corrections Analyzer.Proofread Analyzer.Analyze
                     Analyzer.Lookup Analyzer.FindWord Analyzer.AddEntry Dict.DisplayEntry)
              ;; MACROS that call apply the methods of the analyzer class.
              (FNS AnalyzerFromName Analyzer.CountWords Analyzer.DefaultCorrections Analyzer.DefaultNextWord
                   Analyzer.Name Analyzer.DefaultAddEntry Analyzer.DefaultAnalyze Analyzer.DefaultProofread)
              ;; Functions implementing the default case for various methods of the analyzer class.
              (FNS Analyzer.DefaultLoadWordList Analyzer.DefaultStoreWordList Analyzer.ReadWordList
                   Analyzer.WriteWordList CREATEWORDLISTRDTBL)
              (INITVARS WORDLISTRDTBL)
              (FNS Analyzer.Prop Analyzer.PushProp)
              (MACROS Analyzer.AlphaCharP \Analyzer.TestCorruption Analyzer.Capitalization Analyzer.UCaseP)
              ;; Service MACROS.
              (FNS STREAM.FETCHSTRING)
              (MACROS Stream.Init Stream.NextChar)
              (FNS Analyzer.CorruptWord)
              (GLOBALVARS WORDLISTRDTBL))
       [COMS

;;; TEDIT interface to analyzer.

              (FNS Analyzer.Establish AnalyzerForStream Analyzer.QuitFn Analyzer.BeforeLogout)
              (FNS TEdit.ProofreadMenu PROOFREADER.WHENSELECTEDFN WITH-TEDIT TEdit.Correct TEdit.CountWords
                   TEdit.AddEntry TEdit.Proofread TEdit.SetAnalyzer TEdit.LoadWordList TEdit.StoreWordList
                   Analyzer.TEditMenuItems)
              (INITVARS Analyzer.List Proofreader.AutoCorrect (Proofreader.AutoDelete T)
                     (Proofreader.MenuEdge 'LEFT)
                     Analyzer.TimeProofreader Proofreader.UserFns)
              (GLOBALVARS Analyzer.List Proofreader.AutoCorrect Proofreader.AutoDelete Proofreader.MenuEdge
                     Analyzer.TimeProofreader Proofreader.UserFns)
              (P (Analyzer.TEditMenuItems)
                 (COND ((NOT (FASSOC 'Analyzer.BeforeLogout BEFORELOGOUTFORMS))
                        (push BEFORELOGOUTFORMS '(Analyzer.BeforeLogout]
       (COMS

;;; THE Dict CLASS

              (RECORDS Dict)
              (MACROS Dict.Open Dict.Close Dict.GetEntry Dict.PutEntry Dict.PrintEntry Dict.MapEntries)

;;; utility functions

              (FNS DictFromName Dict.Establish Dict.Prop Dict.Name)
              (INITVARS Dict.DictionaryList)
              (GLOBALVARS Dict.DictionaryList)

;;; a simple dictionary.

              (FNS SimpleDict.New SimpleDict.PutEntry SimpleDict.Lookup SimpleDict.MapEntries
                   SimpleDict.PrintEntries SimpleDict.Test)
              (RECORDS SimpleDict.Node))
       (COMS

;;; the INVERTEDDICT class

              (RECORDS INVERTEDDICT)
              (FNS InvertedDictFromName InvertedDict.Establish InvertedDict.Prop InvertedDict.Name
                   InvertedDict.Open)
              (INITVARS InvertedDict.List)
```

```
            (GLOBALVARS InvertedDict.List))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA InvertedDict.Prop Dict.Prop
                                                                                    Analyzer.Prop])
```

;;; THE ANALYZER CLASS

```
(DECLARE%: EVAL@COMPILE

(DATATYPE Morphalyzer (analyzerName grammar index analyzerProps openFn closeFn proofreadFn analyzeFn lookupFn
                           correctionsFn generateFn conjugateFn findWordFn addEntryFn)
       openFn _ (FUNCTION NILL)
       closeFn _ (FUNCTION NILL)
       proofreadFn _ (FUNCTION Analyzer.DefaultProofread)
       analyzeFn _ (FUNCTION Analyzer.DefaultAnalyze)
       lookupFn _ (FUNCTION NILL)
       correctionsFn _ (FUNCTION Analyzer.DefaultCorrections)
       generateFn _ (FUNCTION NILL)
       conjugateFn _ (FUNCTION NILL)
       findWordFn _ (FUNCTION NILL)
       addEntryFn _ (FUNCTION Analyzer.DefaultAddEntry))
)

(/DECLAREDATATYPE 'Morphalyzer '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                                        POINTER POINTER POINTER POINTER)

       ;; ---field descriptor list elided by lister---

       '28)
```

;; renamed record to avoid a conflict.

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS Analyzer.Open MACRO ((analyzer)
                               (APPLY* (fetch (Morphalyzer openFn) of analyzer)
                                      analyzer)))

(PUTPROPS Analyzer.Close MACRO ((analyzer)
                                (APPLY* (fetch (Morphalyzer closeFn) of analyzer)
                                       analyzer)))

(PUTPROPS Analyzer.Corrections MACRO ((analyzer stream start length)

       (* * returns a list of possible corrections for the string starting at "start" that is "length" characters long.)

                                  (APPLY* (fetch (Morphalyzer correctionsFn) of analyzer)
                                         analyzer stream start length)))

(PUTPROPS Analyzer.Proofread MACRO ((analyzer stream start length prFn)

       (* * The user interface to Analyzer.Analyze.)

                                (APPLY* (fetch (Morphalyzer proofreadFn) of analyzer)
                                       analyzer stream start length prFn)))

(PUTPROPS Analyzer.Analyze MACRO ((analyzer stream start length analFn allowWildCards)

       (* * break up the stream into legal lexical items. calls analFn (analyzer stream start len entries) on each item, where "entries"
       is the analysis of that item. If "entries" is NIL, then the item could not be analyzed.)

                               (APPLY* (fetch (Morphalyzer analyzeFn) of analyzer)
                                      analyzer stream start length analFn allowWildCards)))

(PUTPROPS Analyzer.Lookup MACRO ((analyzer stream start length)

       (* * Look up the substring of stream between start and length in dict.
       "stream" can be a stream, a string, or a list of characters.)

                              (APPLY* (fetch (Morphalyzer lookupFn) of analyzer)
                                     analyzer stream start length)))

(PUTPROPS Analyzer.FindWord MACRO ((analyzer word stream start length)
                                   (APPLY* (fetch (Morphalyzer findWordFn) of analyzer)
                                          analyzer word stream start length)))

(PUTPROPS Analyzer.AddEntry MACRO ((analyzer lemma entry dontRecord)

       (* * add lemma to the dictionary with entry "entry" %. If dontRecord is non-NIL, don't worry about keeping track of this word
       for the word list.)

                                 (APPLY* (fetch (Morphalyzer addEntryFn) of analyzer)
                                        analyzer lemma entry dontRecord)))

(PUTPROPS Dict.DisplayEntry MACRO ((dict entry newwindowflg)
                                   (APPLY* (OR [COND
```

```
                                                ((type? Dict dict)
                                                 (Dict.Prop dict 'DISPLAYENTRYFN))
                                                ((type? INVERTEDDICT dict)
                                                 (InvertedDict.Prop dict 'DISPLAYENTRYFN]
                                            'NILL)
                                       dict entry newwindowflg)))
)
```

;; MACROS that call apply the methods of the analyzer class.

```
(DEFINEQ
```

(AnalyzerFromName

```
  [LAMBDA (dictName remoteName)                                    ; Edited  6-Oct-88 09:56 by jtm:
```

(* * find the analyzer corresponding to the dictionary.)

```
    (PROG (analyzer COLONPOS)
          [COND
             ((NULL dictName)
              (SETQ analyzer (CAR Analyzer.List)))
             [(for i in Analyzer.List do (COND
                                            ([AND (EQ dictName (fetch (Morphalyzer analyzerName) of i))
                                                  (EQ remoteName (Analyzer.Prop i 'RemoteDict]
                                             (SETQ analyzer i)
                                             (RETURN T]
             ((SETQ COLONPOS (STRPOS ":" dictName))
              (SETQ analyzer (AnalyzerFromName (SUBATOM dictName 1 (SUB1 COLONPOS))
                                    (SUBATOM dictName (IPLUS COLONPOS 2)
                                             -1]
          (RETURN analyzer])
```

(Analyzer.CountWords

```
  [LAMBDA (analyzer stream start length)                           (* jtm%: "13-Nov-86 13:32")
    (LET [(n 0)
          (FN (Analyzer.Prop analyzer 'CountWords]
         (COND
            (FN (APPLY* FN analyzer stream start length))
            (T [Analyzer.Analyze analyzer stream start length (FUNCTION (LAMBDA (analyzer stream start length
                                                                                  entries)
                                                                  (add n 1)
                                                                  NIL]
               n])
```

(Analyzer.DefaultCorrections

```
  [LAMBDA (analyzer stream start length)                           (* jtm%: " 7-Apr-87 08:23")
```

(* * returns a list of possible spelling corrections for the given word.)

```
    (PROG [form word wordList caps periods (userDict (Analyzer.Prop analyzer 'UserDict]
          [COND
             [(STREAMP stream)
              (SETFILEPTR stream start)
              (SETQ word (for i from 1 to length collect (BIN stream]
             ((STRINGP stream)
              (SETQ word (for i from 1 to (NCHARS stream) collect (NTHCHARCODE stream i]
          (SETQ caps (Analyzer.Capitalization word))
          (SETQ periods (FMEMB (CHARCODE %.)
                               word))
```

(* * first try transpositions)

```
          (for tail temp on word while (CDR tail) do (SETQ temp (CAR tail))
                                                     (RPLACA tail (CADR tail))
                                                     (RPLACA (CDR tail)
                                                            temp)
                                                     (COND
                                                        ((AND (EQ caps 'FIRST)
                                                              (EQ tail word))
                                                                  (* don't transpose the first letters of a capitalized word.)
                                                         NIL)
                                                        (T (\Analyzer.TestCorruption analyzer word wordList
                                                                  userDict)))
                                                     (RPLACA (CDR tail)
                                                            (CAR tail))
                                                     (RPLACA tail temp))
```

(* * next try deletions)

```
          (COND
             ((CDR word)
              (\Analyzer.TestCorruption analyzer (CDR word)
                     wordList userDict)))
          (for tail temp on word while (CDR tail) do (SETQ temp (CDR tail))
                                                     (RPLACD tail (CDDR tail))
```

```
                                                    (\Analyzer.TestCorruption analyzer word wordList userDict)
                                                    (RPLACD tail temp))

        (* * prepend a character.)

        (SETQ word (CONS (CHARCODE A)
                         word))
        (SELECTQ caps
             (FIRST                                                  (* don't prepend a character before a capitalized word.)
                    NIL)
             (ALL                                                    (* prepend a capital letter.)
                  (for c from (CHARCODE A) to (CHARCODE Z) do (RPLACA word c)
                                                              (\Analyzer.TestCorruption analyzer word wordList
                                                                    userDict)))
             (for c from (CHARCODE a) to (CHARCODE z) do (RPLACA word c)
                                                         (\Analyzer.TestCorruption analyzer word wordList
                                                               userDict)))
        (SETQ word (CDR word))

        (* * insert characters.)

        (for tail on word do (RPLACD tail (CONS (CHARCODE A)
                                                (CDR tail)))
                             [COND
                                ((EQ caps 'ALL)
                                 (for c from (CHARCODE A) to (CHARCODE Z)
                                    do (RPLACA (CDR tail)
                                               c)
                                       (\Analyzer.TestCorruption analyzer word wordList userDict)))
                                (T (for c from (CHARCODE a) to (CHARCODE z)
                                      do (RPLACA (CDR tail)
                                                 c)
                                         (\Analyzer.TestCorruption analyzer word wordList userDict]
                             (COND
                                (periods (RPLACA (CDR tail)
                                                 (CHARCODE %.))
                                         (\Analyzer.TestCorruption analyzer word wordList userDict)))
                             (RPLACD tail (CDDR tail)))

        (* * replace characters)

        (for tail temp on word do (SETQ temp (CAR tail))
                                  [COND
                                     ((OR (EQ caps 'ALL)
                                          (AND (EQ caps 'FIRST)
                                               (EQ tail word)))
                                      (for c from (CHARCODE A) to (CHARCODE Z)
                                         do (COND
                                               ((NEQ temp c)
                                                (RPLACA tail c)
                                                (\Analyzer.TestCorruption analyzer word wordList userDict]
                                  [COND
                                     ((OR (EQ caps NIL)
                                          (NOT (ALPHACHARP (CHCON1 temp)))
                                          (AND (EQ caps 'FIRST)
                                               (NEQ tail word)))
                                      (for c from (CHARCODE a) to (CHARCODE z)
                                         do (COND
                                               ((NEQ temp c)
                                                (RPLACA tail c)
                                                (\Analyzer.TestCorruption analyzer word wordList userDict]
                                  (COND
                                     (periods (RPLACA tail (CHARCODE %.))
                                              (\Analyzer.TestCorruption analyzer word wordList userDict)))
                                  (RPLACA tail temp))
        (SETQ wordList (SORT wordList))
        [for i on wordList do (while (STREQUAL (CAR i)
                                               (CADR i))
                                 do (RPLACD i (CDDR i]
        (RETURN wordList])
```

(Analyzer.DefaultNextWord

```
  [LAMBDA (analyzer stream startPtr searchLength NWFn)          (* jtm%: "29-Oct-85 15:23")

        (* * Scans the stream looking for a word, i.e. a sequence of alphabetic charqacters.
        If the file ptr is already in the middle of such a sequence, it backs up to the beginning of that sequence.
        The function applies NWFn to (stream start stop) for each such word.)

    (SETFILEPTR stream (OR startPtr (SETQ startPtr 0)))
    (bind char end endPtr word length start value quote (filePtr _ (GETFILEPTR stream))
          (EOFPtr _ (GETEOFPTR stream)) first (SETQ endPtr (COND
                                                             (searchLength (IPLUS startPtr searchLength))
                                                             (T EOFPtr)))
                                              (OR (ILEQ endPtr EOFPtr)
                                                  (SETQ endPtr EOFPtr))
       do (SETQ char (AND (ILESSP (GETFILEPTR stream)
```

```
                              endPtr)
                        (BIN stream)))
          (COND
             [(AND char (AND (NUMBERP char)
                             (ILESSP char 128)
                             (Analyzer.AlphaCharP char)))
              (OR start (SETQ start (SUB1 (GETFILEPTR stream]
             (start (SETQ end (GETFILEPTR stream))
                    (SETQ length (IDIFFERENCE end start))
                    (AND char (add length -1))                  (* back up to the last legal char.)
                    [COND
                       (NWFn (SETQ value (APPLY* NWFn analyzer stream start length]
                    (COND
                       ((OR (NULL NWFn)
                            (EQ value T))
                        (RETURN (CONS start length)))
                       (value (RETURN value)))
                    (SETFILEPTR stream end)
                    (SETQ start NIL)))
          (OR char (RETURN])
```

(Analyzer.Name

```
[LAMBDA (analyzer)                                              (* jtm%: "13-Oct-87 10:44")
   (COND
      [(Analyzer.Prop analyzer 'RemoteDict)
       (MKATOM (CONCAT (fetch (Morphalyzer analyzerName) of analyzer)
                       ": "
                       (Analyzer.Prop analyzer 'RemoteDict]
      (T (fetch (Morphalyzer analyzerName) of analyzer])
```

(Analyzer.DefaultAddEntry

```
[LAMBDA (analyzer lemma entry dontRecord errorStream)           (* jtm%: " 7-Apr-87 07:57")
   (LET [(userDict (Analyzer.Prop analyzer 'UserDict]
        (COND
           ((NULL userDict)
            (SETQ userDict (SimpleDict.New))
            (Analyzer.Prop analyzer 'UserDict userDict)))
        (Dict.PutEntry userDict lemma entry)
        (COND
           ((NOT dontRecord)
            (Analyzer.PushProp analyzer 'WordList lemma)))
        lemma])
```

(Analyzer.DefaultAnalyze

```
[LAMBDA (analyzer stream startPtr searchLength NWFn allowWildCards)
                                                                ; Edited 23-Nov-88 08:17 by jtm:

        (* * Scans the stream looking for a word, i.e. a sequence of alphabetic charqacters.
        If the file ptr is already in the middle of such a sequence, it backs up to the beginning of that sequence.
        The function applies NWFn to (stream start stop) for each such word.)

   [COND
      ((STRINGP stream)
       (SETQ stream (OPENSTRINGSTREAM stream]
   (SETFILEPTR stream (OR startPtr (SETQ startPtr 0)))
   (bind char end endPtr length start lookup number initialQuote seprs (userDict _ (Analyzer.Prop analyzer
                                                                                        'UserDict))
        [optSeprCodes _ (OR (Analyzer.Prop analyzer 'OPT-SEPR-CODES)
                            '(39 46 45 47]
        (addAlphaCharCodes _ (Analyzer.Prop analyzer 'ADD-ALPHA-CHAR-CODES))
        (word _ (ALLOCSTRING 100 32))
        (i _ startPtr) first (DECLARE (LOCALVARS . T))
                       [SETQ endPtr (COND
                                       (searchLength (IMIN (GETEOFPTR stream)
                                                           (IPLUS startPtr searchLength)))
                                       (T (GETEOFPTR stream]
      do (SETQ char (AND (add i 1)
                         (ILEQ i endPtr)
                         (BIN stream)))
        (COND
           ((AND start (NUMBERP char)
                 (ILESSP char 128))
            (RPLCHARCODE word (IDIFFERENCE i start)
                   char)))
        [COND
           [[AND char (OR (AND (NUMBERP char)
                               (ILESSP char 128)
                               (Analyzer.AlphaCharP char))
                          (FMEMB char addAlphaCharCodes)
                          (AND allowWildCards (EQ char (CONSTANT (CHARCODE *]
            (COND
               ((NULL start)
                [COND
                   (number (SETQ start (IDIFFERENCE i 2))
```

```
(* we have a number followed by some characters. (e.g. 7th, 21st, etc.) Take in the last digit of the number.)

                  (RPLCHARCODE word 1 number)
                  (SETQ number NIL))
            (T (SETQ start (SUB1 i]
         (RPLCHARCODE word (IDIFFERENCE i start)
                char]
   [(AND char (NUMBERP char)
         (IGEQ char 48)
         (ILEQ char 57))                                    (* a number)
    (COND
       ((NULL start)
        (SETQ number char)
        (SETQ initialQuote NIL))
       (T (RPLCHARCODE word (IDIFFERENCE i start)
                  char]
   ((AND start char (FMEMB char optSeprCodes))              (* optSeprCodes may or may not be a part of the word.)
    (push seprs i))
   [start                                                   (* AND char (add length -1))
                                                            (* back up to the last legal char.)

(* * find the longest string of characters seperated by seprs that the analyzer accepts.)

          (COND
             ((NULL seprs)
              (SETQ seprs i))
             (T (push seprs i)))
          [for stop inside seprs do (SETQ length (SUB1 (IDIFFERENCE stop start)))
                                    (COND
                                       ([SETQ lookup (OR (Analyzer.Lookup analyzer word 0 length)
                                                         (AND userDict (SimpleDict.Lookup userDict word
                                                                                          length]
                                        (RETURN))
                                       ((AND initialQuote (EQP length 1)
                                             (EQ (NTHCHARCODE word 1)
                                                 (CHARCODE s)))
                                        (SETQ lookup 'possessive)
                                        (RETURN]

(* * apply NWFn and return its value if non-NIL.)

          (COND
             ((AND (NULL NWFn)
                   (NEQ lookup 'possessive))
              (RETURN (CONS start length)))
             ((AND (NEQ lookup 'possessive)
                   (SETQ lookup (APPLY* NWFn analyzer stream start length lookup)))
              (RETURN lookup))
             (T (COND
                   ((NEQ i (IPLUS start length 1))          (* we regressed.)
                    (SETQ i (IPLUS start length))

(* don't add 1 so that we will see the quote and initialQuote will get set
("time's"))

                    (SETFILEPTR stream i)                   (* set char to T to prevent the RETURN at the end of the loop.)
                    (SETQ char T)))
                (SETQ start NIL)
                (SETQ seprs NIL)
                (SETQ initialQuote NIL]
   (T (SETQ number NIL)
      (SETQ initialQuote (EQ char (CHARCODE %']
(OR char (RETURN])
```

(Analyzer.DefaultProofread

```
  [LAMBDA (analyzer stream begin length)                    (* jtm%: "16-Dec-87 13:07")
    (PROG (start.length correction startTime stopTime char (n 0))
          (TEDIT.PROMPTPRINT stream "Proofreading . . . " T)

          (* * initialize and back up to the beginning of a word.)

          (SETQ startTime (CLOCK 0))
          (Stream.Init stream begin length)
          [COND
             ((NEQ length 0)
              (while (AND (NUMBERP (SETQ char (BIN stream)))
                          (ALPHACHARP char))
                 do (COND
                       ((EQUAL begin 0)
                        (RETURN))
                       (T (add begin -1)
                          (add length 1)
                          (SETFILEPTR stream begin]

          (* * look for the next spelling error.)
```

```
          [while [SETQ start.length (Analyzer.Analyze analyzer stream begin length (FUNCTION (LAMBDA (analyzer
                                                                                                stream
                                                                                                start
                                                                                                length
                                                                                                entries)
                                                                                            (add n 1)
                                                                                            (COND
                                                                                               ((NULL entries)
                                                                                                (CONS start
                                                                                                      length]
             do

          (* * start.length is a CONS pair of locations delimiting an unrecognizable word.
          Set the selection to it and display it.)

                [COND
                   ((AND Proofreader.UserFns (for fn (word _ (STREAM.FETCHSTRING stream (CAR start.length)
                                                                                (CDR start.length)))
                                             inside Proofreader.UserFns thereis (APPLY* fn word)))
                    (SETQ correction '*SKIP*))
                   (T (TEDIT.SETSEL stream (ADD1 (CAR start.length))
                             (CDR start.length)
                             'RIGHT T)
                      (TEDIT.SHOWSEL stream NIL)
                      (TEDIT.NORMALIZECARET stream)
                      (TEDIT.SHOWSEL stream T)
                      (COND
                         ([NOT (AND Proofreader.AutoCorrect (SETQ correction (TEdit.Correct stream analyzer T]
                          (RETURN]
                (COND
                   [(FMEMB correction '(*SKIP* *INSERT*))
                    [add length (IDIFFERENCE begin (IPLUS (CAR start.length)
                                                          (CDR start.length]
                    (SETQ begin (IPLUS (CAR start.length)
                                       (CDR start.length]
                   ((STRINGP correction)
                    (add length (IDIFFERENCE begin (CAR start.length)))
                                                                        (* move start point.)
                    (add length (IDIFFERENCE (NCHARS correction)
                                       (CDR start.length)))            (* adjust for correction.)
                    (SETQ begin (CAR start.length)))
                   (T (SHOULDNT]
          (SETQ stopTime (CLOCK 0))
          (COND
             (Analyzer.TimeProofreader (TEDIT.PROMPTPRINT stream (CONCAT "Elapsed Time: "
                                                                         (QUOTIENT (DIFFERENCE stopTime startTime
                                                                                               )
                                                                                   1000.0)
                                                                         " seconds.")))
             (start.length (TEDIT.PROMPTPRINT stream "Error found."))
             (T                                                         (* (ADD1 (GETEOFPTR stream)))
                (TEDIT.SETSEL stream (IPLUS begin length 1)
                       0
                       'RIGHT)
                (TEDIT.SHOWSEL stream NIL)
                (TEDIT.NORMALIZECARET stream)
                (TEDIT.SHOWSEL stream T)
                (TEDIT.PROMPTPRINT stream (COND
                                             ((EQUAL n 0)
                                              "No Errors.")
                                             (T (CONCAT n " words proofread.")))
                       T])

)

;; Functions implementing the default case for various methods of the analyzer class.

(DEFINEQ

(Analyzer.DefaultLoadWordList
  [LAMBDA (analyzer file)                                           (* jtm%: "17-Sep-86 09:39")

          (* * adds a word list to the given analyzer.)

    (PROG (wordList)
          (SETQ wordList (Analyzer.ReadWordList file))
          (for i in wordList do (Analyzer.AddEntry analyzer i T T))
          (Analyzer.PushProp analyzer 'WordListFile file])

(Analyzer.DefaultStoreWordList
  [LAMBDA (analyzer file)                                           (* jtm%: "23-Sep-86 09:08")

          (* * adds the current word list to the remote file.)

    (PROG (wordList)
```

```
          (SETQ wordList (Analyzer.Prop analyzer 'WordList))
          [COND
             ((DIRECTORY file)
              (SETQ wordList (APPEND wordList (Analyzer.ReadWordList file]
          (Analyzer.WriteWordList wordList file)
          (Analyzer.PushProp analyzer 'WordListFile file)
          (Analyzer.Prop analyzer 'WordList NIL])
```

(Analyzer.ReadWordList

```
  [LAMBDA (file)                                          ; Edited  9-Mar-89 15:22 by jtm:
    (PROG (firstWord word words stream)
          (SETQ stream (OPENSTREAM file 'INPUT))
          (SETFILEPTR stream 0)
          (SETQ firstWord (READ stream))
          (SETFILEPTR stream 0)
          (COND
             [(LISTP firstWord)                           (* old style format.)
              (RETURN (CDR (READFILE stream]
             (T                                           (* new style format)
                [COND
                   ((NULL WORDLISTRDTBL)
                    (SETQ WORDLISTRDTBL (CREATEWORDLISTRDTBL]
                [while (SKIPSEPRCODES stream WORDLISTRDTBL) do (SETQ word (RSTRING stream WORDLISTRDTBL))
                                                           (COND
                                                              ((EQ 0 (NCHARS word))
                                                               (BIN stream))
                                                              (T (push words word]
                (CLOSEF stream)
                (RETURN words])
```

(Analyzer.WriteWordList

```
  [LAMBDA (wordList file)                                 (* jtm%: "17-Sep-86 10:11")
    (PROG (stream)
          (SETQ stream (OPENSTREAM file 'OUTPUT))
          (SETFILEPTR stream 0)
          (for word in wordList do (printout stream word T))
          (CLOSEF stream])
```

(CREATEWORDLISTRDTBL

```
  [LAMBDA NIL                                             (* jtm%: "17-Sep-86 10:55")
    (LET (RDTBL)
         (SETQ RDTBL (COPYREADTABLE 'ORIG))
         (for SEPR in (GETSEPR RDTBL) do (SETSYNTAX (CHARACTER SEPR)
                                                    'OTHER RDTBL))
         (for BREAK in (GETBRK RDTBL) do (SETSYNTAX (CHARACTER BREAK)
                                                    'OTHER RDTBL))
         (SETSYNTAX (CHARACTER (CHARCODE CR))
                'SEPR RDTBL)
         RDTBL])

)

(RPAQ? WORDLISTRDTBL NIL)

(DEFINEQ
```

(Analyzer.Prop

```
  [LAMBDA a                                               (* jtm%: "13-Oct-87 11:54")
    (LET (p (analyzer (ARG a 1))
            (prop (ARG a 2)))
         (SETQ p (FASSOC prop (fetch (Morphalyzer analyzerProps) of analyzer)))
         (COND
            ((ILEQ a 2)
             (CDR p))
            [p (PROG1 (CDR p)
                      (RPLACD p (ARG a 3)))]
            (T (CDAR (push (fetch (Morphalyzer analyzerProps) of analyzer)
                           (CONS prop (ARG a 3])
```

(Analyzer.PushProp

```
  [LAMBDA (analyzer prop value)                           (* jtm%: "13-Oct-87 10:59")

          (* * pushes value onto a list of values stored at prop.)

    (LET [(prop.values (FASSOC prop (fetch (Morphalyzer analyzerProps) of analyzer]
         (COND
            [(NULL prop.values)
             (push (fetch (Morphalyzer analyzerProps) of analyzer)
                   (CONS prop (LIST value]
            ((NOT (for i in (CDR prop.values) thereis (EQUAL i value)))
             (push (CDR prop.values)
                   value)))
         value])
```

```
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS Analyzer.AlphaCharP MACRO [(CHAR)
                                     (OR (EQ (LRSH CHAR 8)
                                             241)
                                         ([LAMBDA (UCHAR)
                                            (DECLARE (LOCALVARS UCHAR))
                                            (OR (EQ (LRSH UCHAR 8)
                                                    241)
                                                (AND (IGEQ UCHAR (CHARCODE A))
                                                     (ILEQ UCHAR (CHARCODE Z]
                                          (LOGAND CHAR 95])

(PUTPROPS \Analyzer.TestCorruption MACRO [(analyzer word wordList userDict)
                                          (COND
                                             ((OR (Analyzer.Lookup analyzer word)
                                                  (AND userDict (SimpleDict.Lookup userDict word)))
                                              (push wordList (CONCATCODES word])

(PUTPROPS Analyzer.Capitalization MACRO [(word)

          (* * returns NIL, ALL or FIRST)

                                        (COND
                                           ((AND (CAR word)
                                                 (Analyzer.UCaseP (CAR word)))
                                            (COND
                                               ((AND (CADR word)
                                                     (Analyzer.UCaseP (CADR word)))
                                                'ALL)
                                               (T 'FIRST])

(PUTPROPS Analyzer.UCaseP MACRO [(UCHAR)
                                 (OR (AND (IGEQ UCHAR (CHARCODE 361,041))
                                          (ILEQ UCHAR (CHARCODE 361,160)))
                                     (AND (IGEQ UCHAR (CHARCODE A))
                                          (ILEQ UCHAR (CHARCODE Z])
)

;; Service MACROS.

(DEFINEQ

(STREAM.FETCHSTRING
  [LAMBDA (stream start length buffer restorePtr)               (* jtm%: " 3-Apr-87 11:28")
    (LET (pos)
         [COND
            (restorePtr (SETQ pos (GETFILEPTR stream]
         [COND
            ((OR (NULL buffer)
                 (IGREATERP length (NCHARS buffer)))
             (SETQ buffer (ALLOCSTRING length]
         (SETFILEPTR stream start)
         (for i from 1 to length do (RPLCHARCODE buffer i (BIN stream)))
         (COND
            (restorePtr (SETFILEPTR stream pos)))
         (SUBSTRING buffer 1 length])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS Stream.Init MACRO [(stream start length)
                             (COND
                                [(STRINGP stream)
                                 (OR start (SETQ start 0))
                                 (OR length (SETQ length (NCHARS stream]
                                ((NOT (LISTP stream))
                                 (COND
                                    ((NULL start)
                                     (SETQ start 0)))
                                 [COND
                                    ((NULL length)
                                     (SETQ length (IDIFFERENCE (GETEOFPTR stream)
                                                               start]
                                 (SETFILEPTR stream start])

(PUTPROPS Stream.NextChar MACRO [(stream length index)
                                 (COND
                                    ((LISTP stream)
                                     (pop stream))
                                    ((OR (NULL stream)
                                         (ILEQ length 0))
                                     NIL)
```

```
                                    ((STRINGP stream)
                                     (add length -1)
                                     (add index 1)
                                     (NTHCHARCODE stream index))
                                    (T (add length -1)
                                       (BIN stream])
)

(DEFINEQ
```

(Analyzer.CorruptWord

```
  [LAMBDA (analyzer stream start length)                    (* jtm%: " 5-Feb-87 11:23")
```

(* * returns a list of possible spelling corrections for the given word.)

```
    (PROG (form word wordList caps)
          (SETQ word (for i from 1 to length collect (BIN stream)))
          (SETQ caps (Analyzer.Capitalization word))
```

(* * first try transpositions)

```
          (for tail temp on word while (CDR tail) do (SETQ temp (CAR tail))
                                                     (RPLACA tail (CADR tail))
                                                     (RPLACA (CDR tail)
                                                            temp)
                                                     (COND
                                                        ((AND (EQ caps 'FIRST)
                                                              (EQ tail word))
                                                             (* don't transpose the first letters of a capitalized word.)
                                                         NIL)
                                                        (T (\Analyzer.TestCorruption analyzer word wordList)))
                                                     (RPLACA (CDR tail)
                                                            (CAR tail))
                                                     (RPLACA tail temp))
```

(* * next try deletions)

```
          (COND
             ((CDR word)
              (\Analyzer.TestCorruption analyzer (CDR word)
                     wordList)))
          (for tail temp on word while (CDR tail) do (SETQ temp (CDR tail))
                                                     (RPLACD tail (CDDR tail))
                                                     (\Analyzer.TestCorruption analyzer word wordList)
                                                     (RPLACD tail temp))
```

(* * prepend a character.)

```
          (SETQ word (CONS 'A word))
          (SELECTQ caps
              (FIRST                                          (* don't prepend a character before a capitalized word.)
                     NIL)
              (ALL                                            (* prepend a capital letter.)
                   (for c from (CHARCODE A) to (CHARCODE Z) do (RPLACA word c)
                                                              (\Analyzer.TestCorruption analyzer word wordList))
)
              (for c from (CHARCODE a) to (CHARCODE z) do (RPLACA word c)
                                                          (\Analyzer.TestCorruption analyzer word wordList)))
          (SETQ word (CDR word))
```

(* * insert characters.)

```
          (for tail on word do (RPLACD tail (CONS 'A (CDR tail)))
                               [COND
                                  ((EQ caps 'ALL)
                                   (for c from (CHARCODE A) to (CHARCODE Z) do (RPLACA (CDR tail)
                                                                                      c)
                                                                               (\Analyzer.TestCorruption analyzer
                                                                                      word wordList)))
                                  (T (for c from (CHARCODE a) to (CHARCODE z) do (RPLACA (CDR tail)
                                                                                        c)
                                                                                 (\Analyzer.TestCorruption
                                                                                        analyzer word wordList]
                               (RPLACD tail (CDDR tail)))
```

(* * replace characters)

```
          (for tail temp on word do (SETQ temp (CAR tail))
                                    [COND
                                       ((OR (EQ caps 'ALL)
                                            (AND (EQ caps 'FIRST)
                                                 (EQ tail word)))
                                        (for c from (CHARCODE A) to (CHARCODE Z)
                                           do (RPLACA tail c)
                                              (COND
                                                 ((EQ temp c))
                                                 (T (\Analyzer.TestCorruption analyzer word wordList]
```

```
                                    [COND
                                       ((OR (EQ caps NIL)
                                            (NOT (ALPHACHARP temp))
                                            (AND (EQ caps 'FIRST)
                                                 (NEQ tail word)))
                                        (for c from (CHARCODE a) to (CHARCODE z)
                                           do (RPLACA tail c)
                                              (COND
                                                 ((EQ temp (CHARACTER c)))
                                                 (T (\Analyzer.TestCorruption analyzer word wordList]
                                    (RPLACA tail temp))
          (SETQ wordList (SORT wordList))
          [for i on wordList do (while (STREQUAL (CAR i)
                                                 (CADR i))
                                  do (RPLACD i (CDR i]
          (RETURN wordList])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS WORDLISTRDTBL)
)
```

;;; TEDIT interface to analyzer.

```
(DEFINEQ
```

(Analyzer.Establish

```
  [LAMBDA (analyzer)                                        (* jtm%: "13-Oct-87 10:44")
    (AND analyzer (OR (AND (BOUNDP 'Analyzer.List)
                           (bind (analyzerName _ (fetch (Morphalyzer analyzerName) of analyzer)) for tail
                              on Analyzer.List when (EQUAL analyzerName (fetch (Morphalyzer analyzerName)
                                                                               of (CAR tail)))
                              do (RPLACA tail analyzer)
                                 (RETURN T)))
                      (push Analyzer.List analyzer])
```

(AnalyzerForStream

```
  [LAMBDA (stream)                                          (* jtm%: " 2-Oct-85 14:00")

          (* * comment)

    (COND
       ((STREAMPROP stream 'analyzer))
       (T (TEdit.SetAnalyzer stream])
```

(Analyzer.QuitFn

```
  [LAMBDA (window stream textObj)                           (* jtm%: "14-Jan-86 15:58")

          (* * ask the user if he wants to save the word list.)

    (PROG ((analyzer (AnalyzerForStream stream)))
          (COND
             ((AND analyzer (Analyzer.Prop analyzer 'WordList)
                   (STREQUAL "yes" (TEDIT.GETINPUT stream "Do you want to save the word list? " "yes")))
              (TEdit.StoreWordList stream])
```

(Analyzer.BeforeLogout

```
  [LAMBDA NIL                                               (* jtm%: "13-Oct-87 10:45")
    (for analyzer file in Analyzer.List do (COND
                                             ([AND (Analyzer.Prop analyzer 'WordList)
                                                   (EQ 'Y (ASKUSER 10 'N (CONCAT "Do you want to save the word
                                                                                 list for " (fetch (Morphalyzer
                                                                                                    analyzerName)
                                                                                               of analyzer)
                                                                                 "? "]
                                              (COND
                                                 ([NULL (SETQ file (PROMPTFORWORD "Store word list on file:"
                                                                                  (CAR (Analyzer.Prop analyzer
                                                                                               'WordListFile]
                                                  (printout T "Aborted."))
                                                 (T (RESETLST
                                                       (printout T (CONCAT "Storing word list on " file "..."))
                                                       (Analyzer.DefaultStoreWordList analyzer file)
                                                       (printout T "Deleting old version...")
                                                       (RESETSAVE (TTYDISPLAYSTREAM (OPENTEXTSTREAM)))
                                                                  (* to swallow up the output of DIRECTORY)
                                                       (DIRECTORY file '(DELVER)))
                                                    (printout T "done."])

)
```

```
(DEFINEQ
```

(TEdit.ProofreadMenu

```
  [LAMBDA (stream)                                        ; Edited 11-Jul-88 11:02 by jop
```

;;; TEDIT interface to the current analyzer.

```
    (PROG (menuWindow menu analyzer W)
     ;; COND ((WINDOWPROP W (QUOTE DEFWINDOW)) (* so they don't interfere.) (CLOSEW (WINDOWPROP W (QUOTE DEFWINDOW))))
          [SETQ W (CAR (fetch \WINDOW of (TEXTOBJ stream]
          (SETQ menuWindow (WINDOWPROP W 'Proofreader.Menu))
          (COND
             ((NULL menuWindow)
              (SETQ menu (create MENU
                                 ITEMS _ '(("Proofread" Proofread "looks for the next spelling error starting
                                                 from the caret.")
                                           ("Correct" Correct "generates a list of possible corrections for the
                                                 current selection.")
                                           ("Insert" Insert "inserts the selected word into the analyzer's word
                                                 list.")
                                           ("CountWords" Countwords "counts the words in the current selection.")
                                           )
                                 CENTERFLG _ T
                                 MENUOUTLINESIZE _ 2
                                 WHENSELECTEDFN _ (FUNCTION PROOFREADER.WHENSELECTEDFN)))
              (SETQ menuWindow (MENUWINDOW menu T))
              ;; WINDOWPROP menuWindow (QUOTE RESHAPEFN) (QUOTE DON'T)
              ;; (addmenu |menu| |menuWindow|)
              (WINDOWPROP W 'Proofreader.Menu menuWindow)
              (CLOSEW menuWindow)))
          (COND
             ((NOT (OPENWP menuWindow))
              (ATTACHWINDOW menuWindow W Proofreader.MenuEdge 'TOP 'LOCALCLOSE)
              ;; (CAR (WINDOWPROP W (QUOTE PROMPTWINDOW)))
              ;; if you attach the menuWindow to W, then it gets attached to the top-most TEdit menu.
              (OPENW menuWindow)))
          [COND
             ((SETQ analyzer (AnalyzerForStream stream))
              (Analyzer.Proofread analyzer stream (SUB1 (TEDIT.GETPOINT stream]
          (RETURN menuWindow])
```

(PROOFREADER.WHENSELECTEDFN

```
  [LAMBDA (ITEM MENU BUTTON)                              ; Edited 11-Jul-88 10:58 by jop
    (LET [(W (MAINWINDOW (WFROMMENU MENU]
         (SELECTQ (CADR ITEM)
              (Proofread (WITH-TEDIT W (FUNCTION TEdit.Proofread)))
              (Correct (WITH-TEDIT W (FUNCTION TEdit.Correct)))
              (Insert (WITH-TEDIT W (FUNCTION TEdit.AddEntry)))
              (Countwords (WITH-TEDIT W (FUNCTION TEdit.CountWords)))
              NIL])
```

(WITH-TEDIT

```
  [LAMBDA (TEXTOBJ FUNCTION)                              (* jtm%: "30-Mar-87 14:07")
    (LET (EDITOP)
         [COND
            ((WINDOWP TEXTOBJ)
             (SETQ TEXTOBJ (TEXTOBJ TEXTOBJ]
         (COND
            ((SETQ EDITOP (fetch EDITOPACTIVE of TEXTOBJ))
             (TEDIT.PROMPTPRINT TEXTOBJ (CONCAT (COND
                                                   ((EQ EDITOP T)
                                                    "Edit operation")
                                                   (T EDITOP))
                                                " in progress; please wait.")))
            (T (RESETLST
                   [RESETSAVE (\TEDIT.MARKACTIVE TEXTOBJ)
                          '(AND (\TEDIT.MARKINACTIVE OLDVALUE]
                   (replace EDITOPACTIVE of TEXTOBJ with FUNCTION)
                   (APPLY* FUNCTION TEXTOBJ))])
```

(TEdit.Correct

```
  [LAMBDA (stream analyzer autoCorrect)                   (* jtm%: "30-Mar-87 14:09")
    (PROG (selection correction start length items menuWindow)
          [COND
             ((WINDOWP stream)
              (SETQ menuWindow (WINDOWPROP stream 'Proofreader.Menu))
              (SETQ stream (TEXTSTREAM stream)))
             ((type? TEXTOBJ stream)
              (SETQ stream (TEXTSTREAM stream)))
             (T (SETQ menuWindow (WINDOWPROP (CAR (fetch \WINDOW of (TEXTOBJ stream)))
```

```
                                    'Proofreader.Menu]
          (COND
             ([AND (NULL analyzer)
                   (NULL (SETQ analyzer (AnalyzerForStream stream]
              (TEDIT.PROMPTPRINT stream "No analyzer selected." T)
              (RETURN)))
          (SETQ selection (TEDIT.GETSEL stream))
          (SETQ start (fetch (SELECTION CH#) of selection))
          (SETQ length (IDIFFERENCE (fetch (SELECTION CHLIM) of selection)
                              start))
          (TEDIT.PROMPTPRINT stream (CONCAT "Looking for corrections for %"" (STREAM.FETCHSTRING stream
                                                                                (SUB1 start)
                                                                                length)
                                            "%" . . . ")
                T)
          (SETQ items (Analyzer.Corrections analyzer stream (SUB1 start)
                            length))
          [AND autoCorrect (SETQ items (APPEND items '(*INSERT* *SKIP*]
          (COND
             (items [SELECTQ (SETQ correction (MENU (create MENU
                                                            ITEMS _ items
                                                            CENTERFLG _ T
                                                            MENUOFFSET _
                                                            (create POSITION
                                                                    XCOORD _ 6
                                                                    YCOORD _ 6)
                                                            TITLE _ "corrections")))
                            (*INSERT* (TEdit.AddEntry stream analyzer))
                            (*SKIP* NIL)
                            (COND
                               (correction (TEDIT.DELETE stream)
                                       (TEDIT.INSERT stream correction]
                      (TEDIT.PROMPTPRINT stream "" T))
             (T (TEDIT.PROMPTPRINT stream (CONCAT "No corrections for the word %"" (STREAM.FETCHSTRING
                                                                                    stream
                                                                                    (SUB1 start)
                                                                                    length)
                                                  "%".")
                        T)))
          (RETURN correction])
```

(TEdit.CountWords

```
[LAMBDA (stream)                                        (* jtm%: "30-Mar-87 14:11")
  (LET (selection n)
       [COND
          ((OR (WINDOWP stream)
               (type? TEXTOBJ stream))
           (SETQ stream (TEXTSTREAM stream]
       (SETQ selection (TEDIT.GETSEL stream))
       (TEDIT.PROMPTPRINT stream "Counting words in selection . . . " T)
       [SETQ n (Analyzer.CountWords (AnalyzerForStream stream)
                     stream
                     (SUB1 (fetch (SELECTION CH#) of selection))
                     (IDIFFERENCE (fetch (SELECTION CHLIM) of selection)
                            (fetch (SELECTION CH#) of selection]
       [COND
          ((STRINGP n)
           (TEDIT.PROMPTPRINT stream n T))
          (T (TEDIT.PROMPTPRINT stream (CONCAT n " words counted."]
       n])
```

(TEdit.AddEntry

```
[LAMBDA (stream analyzer)                               (* jtm%: "30-Mar-87 14:11")
  (PROG (word)
        [COND
           ((OR (WINDOWP stream)
                (type? TEXTOBJ stream))
            (SETQ stream (TEXTSTREAM stream]
        [COND
           ((NULL analyzer)
            (SETQ analyzer (AnalyzerForStream stream]
        (SETQ word (TEDIT.SEL.AS.STRING stream))
        (COND
           [analyzer (COND
                        ((Analyzer.AddEntry analyzer word T NIL stream)
                         (TEDIT.PROMPTPRINT stream (CONCAT "%"" word "%"" " inserted into local word list.")
                                T))
                        (T (TEDIT.PROMPTPRINT stream "Insert not implemented for this analyzer." T]
           (T (TEDIT.PROMPTPRINT stream "No analyzer selected." T])
```

(TEdit.Proofread

```
[LAMBDA (W)                                             (* jtm%: "16-Dec-87 13:06")
  (LET (sel string (stream (TEXTSTREAM W)))
       (SETQ sel (TEDIT.GETSEL stream))
```

```
         (SETQ string (STREAM.FETCHSTRING stream (SUB1 (fetch (SELECTION CH#) of sel))
                              (fetch (SELECTION DCH) of sel)))
         (COND
            ((STRPOS " " string)                                    (* just analyze the selection.)
             (Analyzer.Proofread (AnalyzerForStream stream)
                    stream
                    (SUB1 (fetch (SELECTION CH#) of sel))
                    (fetch (SELECTION DCH) of sel)))
            (T (Analyzer.Proofread (AnalyzerForStream stream)
                      stream
                      (SUB1 (TEDIT.GETPOINT stream])
```

(TEdit.SetAnalyzer

```
  [LAMBDA (stream analyzer)                                         (* jtm%: "28-Aug-86 09:15")

          (* * sets the analyzer property for the window)

    (PROG (quitFn menuItems)
          [COND
             ((NULL analyzer)
              [SETQ menuItems (for i in Analyzer.List collect (LIST (Analyzer.Name i)
                                                                    (LIST 'QUOTE i)
                                                                    (if (Analyzer.Prop i 'RemoteDict)
                                                                        then "Calls the remote dictionary server"]
              [COND
                 ((NULL menuItems))
                 ((EQ 1 (LENGTH menuItems))
                  (SETQ analyzer (CAR Analyzer.List)))
                 (T (SETQ analyzer (MENU (create MENU
                                                 ITEMS _ menuItems
                                                 TITLE _ "analyzers"
                                                 CENTERFLG _ T]
              (COND
                 ((NULL analyzer)
                  (SETQ analyzer (STREAMPROP stream 'analyzer))
                  (TEDIT.PROMPTPRINT stream (CONCAT "Proofreader is " (AND analyzer (Analyzer.Name analyzer))
                                                    "."))
                        T)
                  (RETURN]
          (TEDIT.PROMPTPRINT stream (CONCAT "Setting proofreader to " (Analyzer.Name analyzer)
                                            "..."))
                T)
          (Analyzer.Open analyzer)
          (STREAMPROP stream 'analyzer analyzer)
          (SETQ quitFn (TEXTPROP stream 'QUITFN))
          [COND
             ((OR (EQ quitFn 'Analyzer.QuitFn)
                  (FMEMB 'Analyzer.QuitFn quitFn))
              NIL)
             ((NULL quitFn)
              (TEXTPROP stream 'QUITFN 'Analyzer.QuitFn))
             (T (TEXTPROP stream 'QUITFN (CONS 'Analyzer.QuitFn quitFn]
                                                                    (* push the function onto the list.)
          (TEDIT.PROMPTPRINT stream "done."))
    analyzer])
```

(TEdit.LoadWordList

```
  [LAMBDA (stream)                                                  (* jtm%: " 9-Oct-85 10:39")

          (* * reads a word list from a remote file and adds it to the given analyzer.)

    (PROG (file (analyzer (AnalyzerForStream stream)))
          (COND
             ((NULL analyzer)
              (TEDIT.PROMPTPRINT stream "Please select a proofreader." T))
             ((NULL (SETQ file (TEDIT.GETINPUT stream "Fetch word list on file: ")))
              (TEDIT.PROMPTPRINT stream "Aborted." T))
             (T (TEDIT.PROMPTPRINT stream (CONCAT "Reading " file "...")
                       T)
                (Analyzer.DefaultLoadWordList analyzer file)
                (TEDIT.PROMPTPRINT stream "done."])
```

(TEdit.StoreWordList

```
  [LAMBDA (stream)                                                  (* jtm%: "28-Jan-87 08:59")

          (* * stores the word list for the given analyzer on a remote file.)

    (PROG (file (analyzer (AnalyzerForStream stream)))
          (COND
             ((NULL analyzer)
              (TEDIT.PROMPTPRINT stream "Please select a proofreader." T))
             ((NULL (Analyzer.Prop analyzer 'WordList))
              (TEDIT.PROMPTPRINT stream "No words to be stored." T))
             ([NULL (SETQ file (TEDIT.GETINPUT stream "Store word list on file: " (CAR (Analyzer.Prop analyzer
```

```
                                                                                    'WordListFile]
            (TEDIT.PROMPTPRINT stream "Aborted." T))
           (T (RESETLST
                  (RESETSAVE (TTYDISPLAYSTREAM (OPENTEXTSTREAM)))
                                                                  (* to swallow up the output of DIRECTORY)
                  (TEDIT.PROMPTPRINT stream (CONCAT "Storing word list on " file "...")
                        T)
                  (Analyzer.DefaultStoreWordList analyzer file)
                  [COND
                     (Proofreader.AutoDelete (TEDIT.PROMPTPRINT stream "Deleting old version..." T)
                            (DIRECTORY file '(DELVER]
                  (TEDIT.PROMPTPRINT stream "done."))])
```

(Analyzer.TEditMenuItems

```
  [LAMBDA NIL                                                    (* jtm%: "23-Oct-87 08:58")
    (AND (BOUNDP 'TEDIT.DEFAULT.MENU)
         (TEDIT.ADD.MENUITEM TEDIT.DEFAULT.MENU
                '(Proofread (FUNCTION TEdit.ProofreadMenu)
                        "Looks for the next spelling error after the caret."
                        (SUBITEMS (Proofread (FUNCTION TEdit.ProofreadMenu)
                                         "Looks for the next spelling error after the caret."
                                         (SUBITEMS (SetProofreader (FUNCTION TEdit.SetAnalyzer)
                                                          "Gives a menu of possible proofreaders to use.")
                                                (LoadWordList (FUNCTION TEdit.LoadWordList)
                                                       "Loads a file of words into the proofreader.")
                                                (StoreWordList (FUNCTION TEdit.StoreWordList)
                                                       "Stores the words added to the proofreader by the user on
                                                       a remote file.")
                                                (AutoCorrect [FUNCTION (LAMBDA (stream)
                                                                           (SETQ Proofreader.AutoCorrect T)
                                                                           (TEDIT.PROMPTPRINT stream
                                                                                  "AutoCorrection is ON." T]
                                                       "The proofreader automatically generates a menu of
                                                       corrections for the user.")
                                                (ManualCorrect [FUNCTION (LAMBDA (stream)
                                                                             (SETQ Proofreader.AutoCorrect)
                                                                             (TEDIT.PROMPTPRINT stream
                                                                                    "AutoCorrection is OFF." T]
                                                       "The user must ask for a menu of corrections from the
                                                       proofreader if he wants one.")))
                                  (Correct (FUNCTION TEdit.Correct)
                                         "generates a list of possible corrections for the current selection.")
                                  (Insert (FUNCTION TEdit.AddEntry)
                                         "inserts the selected word into the analyzer's word list.")
                                  (CountWords (FUNCTION TEdit.CountWords)
                                         "Counts the number of words in the currect selection."])

)

(RPAQ? Analyzer.List NIL)

(RPAQ? Proofreader.AutoCorrect NIL)

(RPAQ? Proofreader.AutoDelete T)

(RPAQ? Proofreader.MenuEdge 'LEFT)

(RPAQ? Analyzer.TimeProofreader NIL)

(RPAQ? Proofreader.UserFns NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS Analyzer.List Proofreader.AutoCorrect Proofreader.AutoDelete Proofreader.MenuEdge
       Analyzer.TimeProofreader Proofreader.UserFns)
)

(Analyzer.TEditMenuItems)

[COND
   ((NOT (FASSOC 'Analyzer.BeforeLogout BEFORELOGOUTFORMS))
    (push BEFORELOGOUTFORMS '(Analyzer.BeforeLogout]


;;; THE Dict CLASS

(DECLARE%: EVAL@COMPILE

(DATATYPE Dict (dictName contents analyzer dictProps subDictionaries openFn closeFn getEntryFn putEntryFn mapFn
                      printEntryFn)
       openFn _ (FUNCTION NILL)
       closeFn _ (FUNCTION NILL)
       getEntryFn _ (FUNCTION NILL)
       putEntryFn _ (FUNCTION NILL)
       mapFn _ (FUNCTION NILL)
       printEntryFn _ (FUNCTION NILL))
```

```
)

(/DECLAREDATATYPE 'Dict '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                          POINTER)
       ;; ---field descriptor list elided by lister---
       '22)

(DECLARE%: EVAL@COMPILE

(PUTPROPS Dict.Open MACRO ((dict)
                           (APPLY* (fetch (Dict openFn) of dict)
                                  dict)))

(PUTPROPS Dict.Close MACRO ((dict)
                            (APPLY* (fetch (Dict closeFn) of dict)
                                   dict)))

(PUTPROPS Dict.GetEntry MACRO ((dict uniqueID prop)
                               (APPLY* (fetch (Dict getEntryFn) of dict)
                                      dict uniqueID prop)))

(PUTPROPS Dict.PutEntry MACRO ((dict uniqueID entry prop)
                               (APPLY* (fetch putEntryFn of dict)
                                      dict uniqueID entry prop)))

(PUTPROPS Dict.PrintEntry MACRO ((dict entry stream)
                                 (APPLY* [COND
                                            ((type? Dict dict)
                                             (fetch (Dict printEntryFn) of dict))
                                            ((type? INVERTEDDICT dict)
                                             (InvertedDict.Prop dict 'PRINTENTRYFN]
                                          dict entry stream)))

(PUTPROPS Dict.MapEntries MACRO ((dict MpFn prop topOnly)        (* MpFn (dict uniqueId entry prop))
                                 (APPLY* (fetch (Dict mapFn) of dict)
                                        dict MpFn prop topOnly)))
)


;;; utility functions

(DEFINEQ

(DictFromName
  [LAMBDA (dictName remoteName)                                  ; Edited  6-Oct-88 09:42 by jtm:
    (PROG (dict COLONPOS)
          [COND
             ((NULL dictName)
              (SETQ dict (CAR Dict.DictionaryList)))
             [(for i in Dict.DictionaryList do (COND
                                                 ([AND (EQ dictName (fetch (Dict dictName) of i))
                                                       (EQ remoteName (Dict.Prop i 'RemoteDict]
                                                  (SETQ dict i)
                                                  (RETURN T]
             [(for i in InvertedDict.List do (COND
                                               ([AND (EQ dictName (fetch (INVERTEDDICT INVERTEDDICTNAME)
                                                                     of i))
                                                     (EQ remoteName (InvertedDict.Prop i 'RemoteDict))
                                                     (SETQ dict (InvertedDict.Prop i 'DICTIONARY]
                                                (RETURN T]
             ((SETQ COLONPOS (STRPOS ":" dictName))
              (SETQ dict (DictFromName (SUBATOM dictName 1 (SUB1 COLONPOS))
                                       (SUBATOM dictName (IPLUS COLONPOS 2)
                                              -1]
          (RETURN dict])

(Dict.Establish
  [LAMBDA (dict)                                                 (* jtm%: "13-Oct-87 10:45")
    (OR (AND (BOUNDP 'Dict.DictionaryList)
             (bind (dictName _ (fetch (Dict dictName) of dict)) for tail on Dict.DictionaryList
                when (EQUAL dictName (fetch (Dict dictName) of (CAR tail))) do (RPLACA tail dict)
                                                                               (RETURN T)))
        (push Dict.DictionaryList dict])

(Dict.Prop
  [LAMBDA a                                                      (* jtm%: "13-Oct-87 11:54")
    (LET (p (dict (ARG a 1))
            (prop (ARG a 2)))
         (SETQ p (FASSOC prop (fetch (Dict dictProps) of dict)))
         (COND
            ((ILEQ a 2)
             (CDR p))
            [p (PROG1 (CDR p)
                      (RPLACD p (ARG a 3)))]
```

```
            (T (CDAR (push (fetch (Dict dictProps) of dict)
                           (CONS prop (ARG a 3])
```

(Dict.Name

```
  [LAMBDA (dict)                                                    (* jtm%: "13-Oct-87 10:45")
    (COND
       [(Dict.Prop dict 'RemoteDict)
        (MKATOM (CONCAT (fetch (Dict dictName) of dict)
                        ": "
                        (Dict.Prop dict 'RemoteDict]
       (T (fetch (Dict dictName) of dict])

)

(RPAQ? Dict.DictionaryList NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS Dict.DictionaryList)
)


;;; a simple dictionary.

(DEFINEQ
```

(SimpleDict.New

```
  [LAMBDA (name)                                                    (* jtm%: "13-Oct-87 10:40")
    (create Dict
            dictName _ name
            getEntryFn _ (FUNCTION SimpleDict.Lookup)
            putEntryFn _ (FUNCTION SimpleDict.PutEntry)
            mapFn _ (FUNCTION SimpleDict.MapEntries)
            contents _ (create SimpleDict.Node])
```

(SimpleDict.PutEntry

```
  [LAMBDA (dict entry value)                                        (* jtm%: " 5-Feb-87 11:29")

          (* * adds the value to dict under entry.)

    (LET (subNode (node (fetch (Dict contents) of dict)))
         [COND
            ((LITATOM entry)
             (SETQ entry (MKSTRING entry]
         (COND
            ((STRINGP entry)
             (for i char from 1 to (NCHARS entry) do (SETQ char (NTHCHAR entry i))
                                                    (COND
                                                       ([NOT (SETQ subNode (FASSOC char (fetch (SimpleDict.Node
                                                                                               subnodes)
                                                                                          of node]
                                                        (SETQ subNode (create SimpleDict.Node
                                                                              char _ char))
                                                        (push (fetch (SimpleDict.Node subnodes) of node)
                                                              subNode)))
                                                    (SETQ node subNode))
             (replace (SimpleDict.Node value) of node with value))
            ((LISTP entry)
             (for char in entry do (COND
                                      ([NOT (SETQ subNode (FASSOC char (fetch (SimpleDict.Node subnodes)
                                                                         of node]
                                       (SETQ subNode (create SimpleDict.Node
                                                             char _ char))
                                       (push (fetch (SimpleDict.Node subnodes) of node)
                                             subNode)))
                                    (SETQ node subNode))
             (replace (SimpleDict.Node value) of node with value)))
          value])
```

(SimpleDict.Lookup

```
  [LAMBDA (dict entry length)                                       (* jtm%: " 7-Apr-87 08:28")

          (* * looks up entry in the dictionary)

    (PROG ((node (fetch (Dict contents) of dict)))
          (COND
             [(OR (STRINGP entry)
                  (LITATOM entry))
              [for i from 1 to (OR length (NCHARS entry)) do (COND
                                                                ([NOT (SETQ node (FASSOC (NTHCHAR entry i)
                                                                                         (fetch (SimpleDict.Node
                                                                                                 subnodes)
                                                                                           of node]
                                                                 (RETURN]
```

```
             (AND node (RETURN (fetch (SimpleDict.Node value) of node]
            ((LISTP entry)
             [for i in entry do (COND
                                   [(AND (NUMBERP i)
                                         (IGREATERP i 9))        (* a character code.)
                                    (COND
                                       ([NOT (SETQ node (FASSOC (CHARACTER i)
                                                                (fetch (SimpleDict.Node subnodes) of node]
                                        (RETURN]
                                   ([NOT (SETQ node (FASSOC i (fetch (SimpleDict.Node subnodes) of node]
                                    (RETURN]
             (AND node (RETURN (fetch (SimpleDict.Node value) of node])
```

(SimpleDict.MapEntries

```
  [LAMBDA (dict fn node path)                                    (* jtm%: "11-Apr-86 15:45")

          (* * maps all of the entries in the dictionary in arbitrary order.)

    [COND
       ((NULL node)
        (SETQ node (fetch (Dict contents) of dict]
    [COND
       ((fetch (SimpleDict.Node value) of node)
        (APPLY* fn dict path (fetch (SimpleDict.Node value) of node]
    (for i in (fetch (SimpleDict.Node subnodes) of node) do (SimpleDict.MapEntries dict fn i
                                                                (APPEND path (LIST (fetch (SimpleDict.Node char)
                                                                                     of i])
```

(SimpleDict.PrintEntries

```
  [LAMBDA (dict stream noValues)                                 (* jtm%: "31-Mar-87 07:37")
    [Dict.MapEntries dict (FUNCTION (LAMBDA (dict entry value)
                                      (COND
                                         (noValues (printout stream (CONCATLIST entry)
                                                                    T))
                                         (T (printout stream (CONCATLIST entry)
                                                             %, value T]
    dict])
```

(SimpleDict.Test

```
  [LAMBDA NIL                                                    (* jtm%: "11-Apr-86 15:49")

          (* * tests the SimpleDict implementation.)

    (LET [(dict (SimpleDict.New 'test]
         (for i in '(asdf asd asdfg asde bfdas) do (Dict.PutEntry dict i i))
         [Dict.MapEntries dict (FUNCTION (LAMBDA (dict entry value)
                                           (printout T entry %, value T]
         dict])

)

(DECLARE%: EVAL@COMPILE

(RECORD SimpleDict.Node (char value . subnodes))
)


;;; the INVERTEDDICT class

(DECLARE%: EVAL@COMPILE

(TYPERECORD INVERTEDDICT (INVERTEDDICTNAME HEADERINDEX KEYINDEX INDEXFILE FILEDIR FILENAME FILEEXT FILEARRAY
                                INVERTEDDICTPROPS))
)

(DEFINEQ
```

(InvertedDictFromName

```
  [LAMBDA (dictName remoteName)                                  ; Edited  6-Oct-88 09:57 by jtm:
    (PROG (dict COLONPOS)
          [COND
             ((NULL dictName)
              (SETQ dict (CAR InvertedDict.List)))
             [(for i in InvertedDict.List do (COND
                                                ([AND (EQ dictName (fetch (INVERTEDDICT INVERTEDDICTNAME)
                                                                      of i))
                                                      (EQ remoteName (InvertedDict.Prop i 'RemoteDict]
                                                 (SETQ dict i)
                                                 (RETURN T]
             ((SETQ COLONPOS (STRPOS ":" dictName))
              (SETQ dict (InvertedDictFromName (SUBATOM dictName 1 (SUB1 COLONPOS))
                                (SUBATOM dictName (IPLUS COLONPOS 2)
                                       -1]
          (RETURN dict])
```

```
(InvertedDict.Establish
  [LAMBDA (dict)                                                    (* jtm%: "13-Oct-87 10:32")
    (OR (bind (name _ (fetch (INVERTEDDICT INVERTEDDICTNAME) of dict)) for tail on InvertedDict.List
           when (EQUAL name (fetch (INVERTEDDICT INVERTEDDICTNAME) of (CAR tail))) do (RPLACA tail dict)
                                                                                    (RETURN T))
        (push InvertedDict.List dict])

(InvertedDict.Prop
  [LAMBDA a                                                         (* jtm%: "13-Oct-87 11:54")
    (LET (p (dict (ARG a 1))
            (prop (ARG a 2)))
         (SETQ p (FASSOC prop (fetch (INVERTEDDICT INVERTEDDICTPROPS) of dict)))
         (COND
            ((ILEQ a 2)
             (CDR p))
            [p (PROG1 (CDR p)
                      (RPLACD p (ARG a 3)))]
            (T (CDAR (push (fetch (INVERTEDDICT INVERTEDDICTPROPS) of dict)
                           (CONS prop (ARG a 3])

(InvertedDict.Name
  [LAMBDA (dict)                                                    (* jtm%: "13-Oct-87 10:33")
    (COND
       [(InvertedDict.Prop dict 'RemoteDict)
        (MKATOM (CONCAT (fetch (INVERTEDDICT INVERTEDDICTNAME) of dict)
                        ": "
                        (InvertedDict.Prop dict 'RemoteDict]
       (T (fetch (INVERTEDDICT INVERTEDDICTNAME) of dict])

(InvertedDict.Open
  [LAMBDA (invertedDict)                                            (* jtm%: " 7-Apr-87 09:01")
    (LET [(OPENFN (InvertedDict.Prop invertedDict 'OPENFN]
         (AND OPENFN (APPLY* OPENFN invertedDict])

)

(RPAQ? InvertedDict.List NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS InvertedDict.List)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA InvertedDict.Prop Dict.Prop Analyzer.Prop)
)

(PUTPROPS ANALYZER COPYRIGHT ("Xerox Corporation" 1985 1986 1987 1988 1989))
```

FUNCTION INDEX

MACRO INDEX

VARIABLE INDEX

RECORD INDEX