<AFFIRM>AFFIRMEXEC..69 30-Sep-81 15:33:22

```
(FILECREATED "26-Sep-81 16:28:33" <AFFIRM>AFFIRMEXEC..69 100505

     changes to:  AddCommandToHistory AddParametersToHistory ProcessCommand AFFIRMEXECCOMS

     previous date: "29-Jun-81 16:49:02" <AFFIRM>AFFIRMEXEC..68)


(PRETTYCOMPRINT AFFIRMEXECCOMS)

(RPAQQ AFFIRMEXECCOMS ((* AFFIRM Executive)
       (FNS AddCommand AddCommandToHistory AddParametersToHistory AffirmExec Annotating AppendChar
            CheckForCurrentType CheckForType ClearTerminalBuffer CloseAnnotation
            DefinedAFFIRMCommand DeleteCommand DetermineObjectClass DTVS Dtvs EatComments
            EndMonitorCycle ErrorInFile GetArgumentsFromList HistoryWarning TopLevel?
            getAssocCommandFcn GetCommand getElements GetParameters IgnoreExcess
            InitializeCommandLoop LoadUserInitializationFile MissingECommand? nodeExpression
            ObjectClassOfParameters OpenAnnotation PerformAutoAppliedCommands PleaseDeclare Plural
            PopTypeStack PrintEnd PrintPrompt ProcessCommand ProcessCommandAbort ReadCommandFile
            readParams ReadRestOfLine ReadSpecialCommandParameters readString RemoveChar
            RestoreTerminalEnvironment SaveTerminalEnvironment SetEventCompletionFlag Singularize
            SkipOverComments SkipToRealChars StartMonitorCycle SuppressAdvisedPrint?
            UpdateLineDeleteMessage ValidateFileName VerbInString)
       (* Command Routines)
       (FNS AffirmCompile AffirmLoad AffirmNews AffirmSave batch EstablishInterLispEnvironment gripe
            limboCommand LowerInterLisp monitor redo renumber RestoreAFFIRMEnvironment save
            setGCmessage setGCpages transcript UserEdit)
       (FNS print printName PrintNews printProofHeader printWhat?)
       (* AFFIRM Executive Tables and Data)
       (RECORDS AFFIRMEnvironment LineDeleteMessages TerminalEnvironment)
       (VARS (Annotating NIL)
             (AnnotationOpen NIL)
             (BatchAnswers NIL)
             (BatchMode NIL)
             (BootStrapping NIL)
             (CommandTerminator ';)
             (CurrentTerminalEnvironment NIL)
             (DeleteControlTypes ' (LINEDELETE 1STCHDEL NTHCHDEL POSTCHDEL EMPTYCHDEL))
             (FreezeFileName 'Frozen-AFFIRM)
             (GripeSubjectMaxLength 36)
             (HistorylessCommands (QUOTE (stop abort exec quit)))
             (ImplicitE NIL)
             (LineDeleteMessage (create LineDeleteMessages CommandEnv← (DELETECONTROL 'LINEDELETE)
                                        NormalEnv←
                                        (DELETECONTROL 'LINEDELETE)
                                        ParameterEnv← "<
~>"))
             (LispCommands ' (e ImplicitE))
             (LispFormStarters <LeftSqBracket LeftParenthesis>)
             (MessageNoTranscriptGiven NIL)
             (OldBreakAccess NIL)
             (OriginalCommandWord NIL)
             (OwnReadCommands <CommandTerminator>)
             (PascalReadTable (COPYREADTABLE 'ORIG))
             (SpecialReadCommands ' (@ e fix ImplicitE note profile redo stop undo))
             (Tab 7)
             (Terminal T)
             (UnBufferedCommands ' ((invoke let put replace suppose)
                                    (apply augment axiom discard eval lemma schema try use)))
             (UsingTed NIL))
       (VARS (CurrentBreakCharacters (APPEND (GETBRK PascalReadTable)
                                             (GETSEPR PascalReadTable)))
             (NormalPrompt "U:")
             (BatchErrorPrompt (CONCAT "##" Blank NormalPrompt))
             (Prompt NormalPrompt))
       (P (SETBRK '
                  (4 5 6 14 16 17 18 19 20 21 28 29 34 30 33 38 40 41 42 43 44 45 46 47 58 59 60 61
                     62 64 91 93 94 123 124 125 126)
                  NIL PascalReadTable))
       [IFPROP AFFIRMCommand * (for c in (fetch Commands of PredefinedNames)
                                  collect
                                  (COND ((NLISTP c)
                                         c)
                                        (T (CAR c]
       (* Redefined INTERLISP functions)
       (FNS AffirmGCGAG AffirmPRINTBELLS)
       (P (if (FGETD 'InterLispGCGAG)
              else
              (MOVD 'GCGAG 'InterLispGCGAG))
          (if (FGETD 'InterLispSort)
              else
              (MOVD 'SORT 'InterLispSort))
```

```
          (if (FGETD 'InterLispPRINTBELLS)
              else
              (MOVD 'PRINTBELLS 'InterLispPRINTBELLS))
          (MOVD 'AffirmGCGAG 'GCGAG)
          (MOVD 'AffirmSORT 'SORT)
          (MOVD 'AffirmPRINTBELLS 'PRINTBELLS))
        (* File / File Name Handling)
        (FNS getFileName makeFileName markFileNameAsUsed OKtoUseCOMS?)))
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* AFFIRM Executive) ]

(DEFINEQ

                                                                    1

(AddCommand
  [LAMBDA (casedCommands dontPrint)                    (* D.Thompson " 6-Feb-80 09:00")

        (* * This routine adds commands to the command list. It also adds the AFFIRMCommand property to the property list of
        the command name atom.)


    (PROG (addedCommands commandName cmdNameChars)
          (until casedCommands do (printout NIL .TAB0 0 "what's the command name? ")
                                  (casedCommands←(READ T)))
          (for c in (MKLIST casedCommands)
             do (if (LITATOM c)
                    then commandName←(L-CASE c)
                         (printout NIL T DoubleQuote commandName DoubleQuote .)
                         (if commandName MEMB KnownNames:Commands
                             then (printout NIL "is already a defined command name." T)
                           else KnownNames:Commands←(MERGEINSERT commandName
                                                                 (MKLIST KnownNames:Commands)
                                                                 T)
                                (PUTPROP commandName 'AFFIRMCommand commandName)
                                (printout NIL "added to the command name list." T))
                         cmdNameChars←(CHCON commandName)
                         (if (EQLENGTH cmdNameChars 1) and (cmdNameChars:1 lt (CHCON1 'a)
                                                                   or cmdNameChars:1 gt
                                                                      (CHCON1 'z))
                               and cmdNameChars:1 ~MEMB (GETBRK PascalReadTable)
                             then (printout NIL .TAB0 0 "WARNING:" , DoubleQuote c DoubleQuote ,
                                        "is not a break character in PascalReadTable!!"
                                        T))
                         addedCommands← < !! addedCommands commandName>
                  else (printout NIL T c , "is not a legal command form!")))
          (RETURN (if addedCommands
                      then <'AddCommand (KWOTE (if addedCommands::1
                                                   then addedCommands
                                                 else addedCommands:1))
                           T>
                    else <'NILL >])

                                                                    2

(AddCommandToHistory
  [LAMBDA (command commandFile)                        (* R.Erickson "25-Sep-81 21:09")

        (* * This routine adds a command to the history list. The parameters are defaulted to NIL at this point, and will be
        added later. The AFFIRM escape-to-lisp commands E and IMPLICITE are NOT added, since lisp will do that itself.)


    (if LISPXHISTORY
        then (if commandFile=Terminal or PRINTCOMMAND
                 then (if command MEMB LispCommands
                          elseif command MEMB HistorylessCommands
                            then                                  (* don't record things like stop;
                                                                  or exec; so that fix; will pick the right one)
                                 NIL
                          else LISPXHIST←(HISTORYSAVE LISPXHISTORY Prompt NIL command
                                                      (if command MEMB OwnReadCommands
                                                          then NIL
                                                        else <CommandTerminator>])

                                                                    3

(AddParametersToHistory
  [LAMBDA (command commandFile parameters replaceWhatsThere)
                                                       (* R.Erickson "22-Jul-81 17:44")

        (* * This routine adds a command's parameters to the history list. The parameters are in their original state: no
        checking has been done. The AFFIRM escape-to-lisp commands E and IMPLICITE are NOT added, since lisp will do that.)


    [if LISPXHISTORY and (LISTP LISPXHIST)
        then (if commandFile=Terminal or PRINTCOMMAND
                 then (if command MEMB LispCommands
```

```
                    elseif command MEMB HistorylessCommands
                      then                                  (* don't record things like stop;
                                                            or exec; so that fix; will pick the right one)
                    else (LISPXHIST:1::1←(if replaceWhatsThere
                                             then (if command MEMB OwnReadCommands
                                                     then parameters
                                                   else < ! parameters CommandTerminator>)
                                           else < !(MKLIST LISPXHIST:1::1) !(MKLIST parameters)
                                                >]
    (if PRINTCOMMAND and commandFile~=Terminal
        then (if command MEMB LispCommands
                  then (printout NIL "e " .PPV parameters T)
                else (PrettyPrintCommand <command ! parameters
                                           !(if command MEMB OwnReadCommands
                                                then NIL
                                              else <CommandTerminator>)
                                           >
                                         NIL PascalReadTable])
```

4

```
(AffirmExec
  [LAMBDA NIL                                           (* D.Thompson "29-Aug-80 11:53")

        (* * This routine is the main entry point for AFFIRM. -
        It initializes the command loop and then enters it. -
        If the command loop breaks back to here, we simply iterate (until the user gets tired.))


    (while T do (InitializeCommandLoop)
                (DTVS 'TopOfDtvs T])
```

5

```
(Annotating
  [LAMBDA NIL                                           (* D.Thompson " 8-Oct-79 09:14")
    Annotating←(UserProfile 'AnnotatingTranscript T)
    AnnotationOpen←NIL])
```

6

```
(AppendChar
  [LAMBDA (atom character duplicate)                    (* D.Thompson "24-Nov-79 23:56")

        (* * This routine appends the character provided as the second parameter onto the end of the atom provided as the
        first parameter, using the Boolean flag provided as the third parameter to decide whether to unconditionally add the
        character, or to check to see if the atom already ends in the character, thus avoiding doubling the character.)


    (if duplicate or character~=(NTHCHAR atom -1)
        then (PACK* atom character)
      else atom])
```

7

```
(CheckForCurrentType
  [LAMBDA NIL                                           (* edited: "11-SEP-78 19:05")
    (if ~CurrentType
        then (PRINTLINES T "No current type!!")
             (ERROR!])
```

8

```
(CheckForType
  [LAMBDA (typeName dontTellUser noType)                (* D.Thompson "28-Aug-80 11:53")

        (* * This routine performs spelling correction on type names. If the type name SHOULD be a type name but isn't
        currently defined, we try to define it, either via needs (finding it somewhere and loading or reading), or by
        minimally specifying it.)


    (if (LISTP typeName)
        then typeName←(PACK typeName))
    (if typeName=T
        then (if CurrentType
                 else (AffirmError "There isn't any current type."))
```

```
    elseif (AFFIRMSpellingCorrect typeName KnownNames:Types NIL T dontTellUser)
    else (SELECTQ (L-CASE noType)
                  (ok NIL)
                  [declare (PROG (keyList optionList question response)
                                 (ValidateFileName typeName 'Types T)
                                 (printout NIL .TAB0 0 "Type" , typeName ,
                                           "has not yet been specified."
                                           T)
                                 (optionList←NIL)
                                 (question←
                    "Should the spec be found via needs, or should it be minimally specified? ")
                                 (keyList←'((Find " its specification via needs")
                                    (Needs NIL RETURN 'Find)
                                    (Specify " the type (minimally)")
                                    (Minimal " specification" RETURN 'Specify)
                                    (No " (neither one: abort command)")))
                                 (response←(AFFIRMUSER NIL 'Specify question keyList NIL NIL
                                                       optionList))
                                 (RETURN (SELECTQ response
                                                  ((Abort No)
                                                    (AffirmError))
                                                  (Specify (MinimalTypeSpec typeName T))
                                                  (Find (needs 'Types typeName CurrentType)
                                                        (if (Loaded? typeName 'Types)
                                                          else (MinimalTypeSpec typeName)))
                                                  (PROGN (Unexpected 'ImplicitNeed)
                                                         (MinimalTypeSpec typeName]
                  (nil (AffirmError <typeName "isn't a defined type." >))
                  (Unexpected 'TypeDisposition T])
```

9

(ClearTerminalBuffer

```
  [LAMBDA (commandFile)                                    (* D.Thompson "11-Feb-80 12:58")

        (* * This routine clears and restores the terminal input buffer, in the process avoiding an Interlisp bug.)


    (PROG (buffer)
          (if commandFile=Terminal
              then (PRINTBELLS)
                   (DOBE)
                   (CLEARBUF T T)
                   (if buffer=(LINBUF T)
                       then (BKLINBUF buffer))
                   (if buffer←(SYSBUF T)
                       then (BKSYSBUF buffer])
```

10

(CloseAnnotation

```
  [LAMBDA NIL                                              (* D.Thompson "24-Jul-80 16:45")

        (* * This routine outputs the closing bracket on the command line environment when we're annotating.)


    (if Annotating and AnnotationOpen
        then (Dprintout NIL RightSqBracket AtSign Asterisk T)
             AnnotationOpen←NIL])
```

11

(DefinedAFFIRMCommand

```
  [LAMBDA (command)                                        (* D.Thompson " 5-Nov-79 13:35")

        (* * This routine uses the property list of an atom to determine whether or not the atom is an AFFIRM command.)


    (if (GETPROP command 'AFFIRMCommand)
        then T
      else NIL])
```

12

(DeleteCommand

```
  [LAMBDA (casedCommands dontPrint)                        (* D.Thompson " 6-Feb-80 09:15")
```

```
(* * This routine removes commands from the command list. It is smart enough to handle the removal of synonyms.
It also removes the AFFIRMCommand property from the property list of the command name atom.)


(PROG (commandName deletedCommands synonym)
      (if KnownNames:Commands
        else (printout T .TAB0 0 "The command name list is empty!!" T)
             (RETURN NIL))
      (until casedCommands do (printout NIL .TAB0 0 "what's the command name? ")
                              (casedCommands←(READ T)))
      [for c in (MKLIST casedCommands)
         do (if (LITATOM c)
                then commandName←(L-CASE c)
                     (if synonym←(SpellingSynonym commandName KnownNames:Commands)
                         then KnownNames:Commands←(REMOVE (if synonym=commandName
                                                              then commandName
                                                            else <commandName ! synonym>)
                                                          KnownNames:Commands)
                              (REMPROP commandName 'AFFIRMCommand)
                              deletedCommands← < !! deletedCommands commandName>
                              (if dontPrint
                                else (printout NIL .TAB0 0 DoubleQuote commandName DoubleQuote ,
                                             "removed from the command name list."
                                             T))
                       else (if dontPrint
                              else (printout NIL .TAB0 0 DoubleQuote c DoubleQuote ,
                                           "isn't a defined command."
                                           T)))
              else (if dontPrint
                     else (printout NIL .TAB0 0 c , "is not a legal command form!"]
      (RETURN (if deletedCommands
                  then <'DeleteCommand (KWOTE (if deletedCommands::1
                                                  then deletedCommands
                                                else deletedCommands:1))
                       T>
                else <'NILL >])
```

13

(DetermineObjectClass

```
[LAMBDA (element objectClasses dontAskUser dontTellUser)
                                                    (* D.Thompson "28-Aug-80 11:51")

(* * This routine figures out the object class from the element by spelling-correcting the element through the list
of possible object classes.)


(PROG (default keyList question object objects optionList)
      (objectClasses←(if objectClasses
                         then (MKLIST objectClasses)
                       else (getElements 'AFFIRMObjects)))
      (objects←(for object in objectClasses when (AFFIRMSpellingCorrect element
                                                                        (getElements object)
                                                                        NIL T dontTellUser)
                  collect object))
      [if objects
          then (if (EQLENGTH objects 1)
                   then object←objects:1
                 else (if dontTellUser
                        else (printout NIL .TAB0 0 element , "is currently a member of" , #
                                       (AFFIRMMAPRINT NIL objects T T)))
                      (if dontAskUser
                          then object←NIL
                        else question←"Which one do you want here? " keyList←
                             <'(/" (None!)") !(for x in objects collect <x NIL>) > optionList←NIL
                             default←(if 'Types MEMB objects
                                         then 'Types
                                       else objects:1)
                             (if object←(AFFIRMUSER NIL default question keyList NIL NIL
                                                    optionList)='/
                                 then object←NIL]
      (RETURN object])
```

14

(DTVS

```
[LAMBDA (commandFile dontPrint)                     (* D.Thompson " 5-Sep-80 19:41")

(* * This routine acts as the ReadCommandFile executive. It opens the file and then calls Dtvs to actually read and
```

execute the commands in the file.)

```
(PROG (oldinputfile PRINTCOMMAND (ERRORTYPELST (<'(16 (PROGN BREAKCHK←NIL
                                                             PRINTMSG←NIL
                                                             (RETFROM 'Dtvs NIL T)))
                                                    ! ERRORTYPELST>))
                    STOPAFFIRM)
      (PromptPrinted←NIL)
  TopOfDtvs
      (STOPAFFIRM←NIL)
      (PRINTCOMMAND←NIL)
      (if commandFile='TopOfDtvs
          then (if UsingTed=NIL
                   then commandFile←Terminal
                 else (if commandFile←(INFILEP TEDFILE)
                          then STOPAFFIRM←T
                               PRINTCOMMAND←T
                        else (printout NIL .TAB0 0 "Can't find file" , TEDFILE Period T)
                             commandFile←Terminal)))
      (if commandFile←(getFileName commandFile NIL "File of commands: ")
          then (if dontPrint or PRINTCOMMAND
                 else (printout NIL .TAB0 0 LeftParenthesis "Reading AFFIRM commands from" ,
                              commandFile RightParenthesis T))
               (if commandFile~=Terminal
                   then (CLOSEF? commandFile))
               oldinputfile←(INFILE commandFile)
               (Dtvs commandFile)
        else (AffirmError))
      (if STOPAFFIRM
          then commandFile←'TopOfDtvs
               (if (INPUT)
                   ~=Terminal
                   then (CLOSEF? (INPUT Terminal)))
               (LOGOUT)
               PromptPrinted←T
               (GO TopOfDtvs))
      (RETURN (CLOSEF? (INFILE oldinputfile])
```

15

```
(Dtvs
  [LAMBDA (commandFile)                                    (* D.Thompson "10-Sep-80 12:55")
```

(* This routine executes commands from the file provided as its parameter, until control is sneakily revoked by a RETFROM when the file's commands are exhausted.)*

```
    (PROG (abortCondition AutoInfix command CurrentCommand currentTimingMark (LowerExec (LowerExec+1))
                          parameters result ADDSPELLFLG)
          (while T do (if (NLSETQ (while T
                                     do (command←NIL)
                                        (CurrentCommand←NIL)
                                        (parameters←NIL)
                                        (result←NIL)
                                        (abortCondition←'input)
                                        (PagesWarning)
                                        (currentTimingMark←(StartMonitorCycle))
                                        (PrintPrompt commandFile LowerExec)
                                        (OpenAnnotation commandFile)
                                        (command←(GetCommand commandFile))
                                        (AddCommandToHistory command commandFile)
                                        (CurrentCommand←command)
                                        (parameters←(GetParameters command commandFile))
                                        (AddParametersToHistory command commandFile parameters T)
                                        (CloseAnnotation)
                                        (PrintEnd command commandFile)
                                        (abortCondition←'processing)
                                        (CHECKNIL)
                                        (if (UNDONLSETQ (PROGN result←(ProcessCommand command
                                                                                      parameters
                                                                                      commandFile NIL)
                                                                       abortCondition←'finished))
                                          else (ProcessCommandAbort command parameters commandFile T))
                                        (CHECKNIL)
                                        (PerformAutoAppliedCommands command result abortCondition)
                                        (SetEventCompletionFlag command commandFile)
                                        (EndMonitorCycle currentTimingMark command)))
                        else (ProcessCommandAbort command parameters commandFile abortCondition])
```

16

```
(EatComments
  [LAMBDA (list)                                        (* D.Thompson " 2-Sep-80 11:50")
    (PROG (processed tail)
          (list←(MKLIST list))
          (while tail←(LeftCurlyBracket MEMB list) do (processed← < !! processed !(LDIFF list tail)
                                                                  >)
                                                      (if list←(RightCurlyBracket MEMB tail)
                                                          then list←list::1
                                                        else (AffirmError
                                                                <"Ill-formed comment:" tail>)))
          (RETURN < !! processed ! list>])
```

17

```
(EndMonitorCycle
  [LAMBDA (previousTimingMark associatedCommand)        (* D.Thompson "25-Sep-79 13:19")
    (PROG (currentTimingMark elapsedTime)
          (if associatedCommand=NIL
              then associatedCommand←"previous command")
          (currentTimingMark←(CLOCK 2))
          (if Timer and (FIXP previousTimingMark)
              then elapsedTime←currentTimingMark-previousTimingMark
                   (printout NIL .TAB0 0 "Timing for" , associatedCommand ":" ,
                             (if (IGEQ elapsedTime 0)
                                 then (QUOTIENT elapsedTime 1000.0)
                               else "(? negative! ?)")
                             , "seconds." T])
```

18

```
(ErrorInFile
  [LAMBDA (command parameters)                          (* D.Thompson " 3-Sep-80 19:32")
    (if [NLSETQ (PROGN (PRINTBELLS)
                       (DOBE)
                       (CLEARBUF T T)
                       (printout NIL .TAB0 0 "The command line is" , #
                                 (PrettyPrintCommand
                                   <command ! parameters
                                     !(if command MEMB OwnReadCommands
                                          then NIL
                                        else <CommandTerminator>)
                                   >
                                   NIL PascalReadTable)
                                 T "(Type  ok;  to continue reading from the file;" T
                                 " Type  stop;  to ignore the rest of the file.)"
                                 T)
                       (if (Dtvs Terminal)='aborted
                           then (ERROR!]
        then T
      else NIL])
```

19

```
(GetArgumentsFromList
  [LAMBDA (list pack alreadySeparated)                  (* D.Thompson " 9-Sep-80 16:28")

          (* * This routine breaks up a list into a sequence of parameters, where a comma is assumed as the separator.)


    (PROG (arguments args done)
          (if alreadySeparated
              then (RETURN list)
            elseif list
              then list←(MKLIST list)
                   arguments←(until done collect (if args←(Comma MEMB list)
                                                     then (PROG1 (LDIFF list args)
                                                                 list←args::1)
                                                   else done←T
                                                        list))
                   [RETURN (if pack
                               then (for a in arguments collect (PACK a))
                             else (for a in arguments collect (if (EQLENGTH a 1)
                                                                  then a:1
                                                                else a]
            else (RETURN NIL])
```

20

```
(HistoryWarning
  [LAMBDA NIL                                          (* D.Thompson "17-Mar-80 18:50")
    (printout NIL .TAB0 0 "LISPXHIST is NIL. Please notify AFFIRM personnel via a GRIPE." T
              "Please send along your transcript."
              T])
```

21

```
(TopLevel?
  [LAMBDA NIL                                          (* D.Thompson "29-Aug-80 11:46")
    LowerExec←0])
```

22

```
(getAssocCommandFcn
  [LAMBDA (commandName checkArgs)                      (* D.Thompson "27-Jun-79 14:43")
    (PROG (args)
          (if (FGETD commandName)
              then args←(ARGLIST commandName)
                   (printout T "It has an associated function" T commandName)
                   (MAPRINT args T "(" ")" ", ")
                   (printout T "." T)
                   (if checkArgs and args:1~='file
                       then (printout T .TAB0 0
                                      "WARNING: the first parameter must be named %"file%" !!"
                                      T))
            else (printout T "It has no associated function." T])
```

23

```
(GetCommand
  [LAMBDA (commandFile)                                (* D.Thompson " 5-Mar-80 18:31")

          (* * This routine reads a command name from the command file.)


    (PROG (command)
          (CONTROL ~(UserProfile 'CommandLineBuffering T))
          (UpdateLineDeleteMessage 'Command)
          (SkipOverComments commandFile)
          (SkipToRealChars commandFile)
          (if ImplicitE and (PEEKC commandFile) MEMB LispFormStarters
              then OriginalCommandWord←NIL
                   command←'ImplicitE
                   (printout NIL "(ImplicitE:)" ,)
            else OriginalCommandWord←(RATOM commandFile PascalReadTable)
                 command←(if (SpellingSynonym OriginalCommandWord KnownNames:Commands)
                             elseif ImplicitE and (MissingECommand? OriginalCommandWord commandFile)
                               then 'ImplicitE
                             elseif (AFFIRMSpellingCorrect (L-CASE OriginalCommandWord)
                                                           KnownNames:Commands)
                             else (ClearTerminalBuffer commandFile)
                                  (printout NIL .TAB0 0 "Using 'note'; skipping to semicolon..." T)
                                  'note))
          (if command MEMB LispCommands
              then                                     (* disable line buffering)
                   (CONTROL T)
            elseif (GETCONTROL)
              then                                     (* line buffering disabled: re-enable unless the
                                                       specific command wants it disabled.)
                   (if command ~MEMB UnBufferedCommands
                       then (CONTROL)))
          (RETURN command])
```

24

```
(getElements
  [LAMBDA (object Names)                               (* D.Thompson "21-Oct-80 17:22")

          (* * This routine returns the elements of the object class provided as the parameter.)


    Names←(if Names
```

```
        else KnownNames)
  (SELECTQ object
          (AFFIRMObjects Names:AFFIRMObjects)
          (Arcs Names:Arcs)
          (Axioms NIL)
          (Commands Names:Commands)
          (Definitions NIL)
          (Directories Names:Directories)
          (DiscardOptions Names:DiscardOptions)
          (Disconnected NIL)
          (Files Names:Files)
          (FileTypes Names:FileTypes)
          (Groups Names:Groups)
          (HelpTopics Names:HelpTopics)
          (History NIL)
          (Interfaces NIL)
          (Lemmas NIL)
          (Lhs NIL)
          (Nodes Names:Nodes)
          (PrintObjects Names:PrintObjects)
          (ProfileEntries Names:ProfileEntries)
          (Schemas NIL)
          (Types Names:Types)
          (TypeParts Names:TypeParts)
          (Variables NIL)
          (Unexpected Names:Unexpected)
          (Unexpected 'AFFIRMObject])
```

25

```
(GetParameters
  [LAMBDA (command commandFile)                    (* D.Thompson " 2-Sep-80 12:04")

        (* * This routine reads the parameters for each of AFFIRM's command functions (unless the command is explicitly on
        the no-read list). -
        Several commands get special treatment, but most are simply read-to-end-of-line.)


    (UpdateLineDeleteMessage 'Parameters)
    (if command MEMB OwnReadCommands
        then NIL
      elseif command MEMB SpecialReadCommands
        then (ReadSpecialCommandParameters command commandFile)
      else (ReadRestOfLine commandFile T])
```

26

```
(IgnoreExcess
  [LAMBDA (list)                                   (* D.Thompson "28-Aug-80 12:41")
    (if list
        then (printout NIL .TAB0 0 LeftParenthesis Ellipses , "Ignoring excess parameters" ,
                       DoubleQuote .PPVTL (MKLIST list)
                       DoubleQuote Period RightParenthesis T])
```

27

```
(InitializeCommandLoop
  [LAMBDA NIL                                      (* D.Thompson " 3-Sep-80 14:38")
    (if CurrentTerminalEnvironment=NIL
        then (SaveTerminalEnvironment))
    (RestoreTerminalEnvironment)
    (if (EVALV CurrentType)='NOBIND or (EVALV CurrentType)=NIL
        then (Edit 'Basis))
    LowerExec←-1
    oldinputfile←Terminal
    (CLOSEALL])
```

28

```
(LoadUserInitializationFile
  [LAMBDA NIL                                      (* D.Thompson " 5-Sep-80 20:10")

        (* * This routine loads the user's initialization file if one exists.)


    (PROG (fileName)
          (if fileName←(getFileName (UserProfile 'UserInitializationFileName))
```

```
        then (printout NIL .TAB0 0 "Loading initialization file..." T)
             (ERSETQ (AffirmLoad fileName])
```

29

```
(MissingECommand?
  [LAMBDA (firstAtom commandFile)                    (* D.Thompson "28-Aug-80 11:52")
    (if (UserProfile 'ImplicitECommand T) and commandFile=Terminal
        and (firstAtom MEMB <LeftParenthesis LeftSqBracket> or 'NOBIND ~=(EVALV firstAtom)
                       or (FGETD firstAtom) and (ARGLIST firstAtom):1~='file
                       and (GETPROP firstAtom 'AFFIRMCommand)=NIL
                       or (FASSOC firstAtom LISPXMACROS) or firstAtom MEMB CLISPFORWORDSPLST
                       or firstAtom MEMB CLISPIFWORDSPLST)
      then T
    else NIL])
```

30

```
(nodeExpression
  [LAMBDA (input)                                    (* R.Erickson "27-Oct-80 16:51")

        (* * Returns a node from input. Allows the user to christen a node at the same time it is being given as an
        expression. The node is returned. -
        The parameter is a just-parsed expression sequence.)


    (PROG (name node expr)
          (if input
              then (if input::1
                       then name←input:1

          (* if assigning a name, we insist that the thing being christened be put thru the interface mechanism
          (and thus not be a name itself))


                       (pop input))
                   expr←(PexecLists input:1 T name)
                   (if (OccursAsOperatorIn IH1OP expr)
                       then (AffirmError "IH is not meaningful in a theorem."))
                   node←(ExprToNode expr)
                   (if name
                       then (SetName name node))
                   (RETURN node)
            else (AffirmError])
```

31

```
(ObjectClassOfParameters
  [LAMBDA (parameters objectClasses internalCall)    (* D.Thompson "10-Sep-80 13:05")

        (* * This routine takes a parameter list and -
        (1) determines the object class of the first parameter; -
        (2) breaks the atom sequence into an argument sequence, assuming comma is the argument separator.)


    (PROG (element object)
          (if parameters
            else (AffirmError "Expecting an object class and a sequence of objects!"))
          (if object←(AFFIRMSpellingCorrect parameters:1 KnownNames:AFFIRMObjects NIL T NIL)
              then ParameterElements←(GetArgumentsFromList parameters::1 NIL internalCall)
                   (if ParameterElements
                     else (AffirmError (CONCAT "You didn't supply any " (L-CASE object)
                                               Period)))
            else ParameterElements←(GetArgumentsFromList parameters NIL internalCall)
                 (if ParameterElements
                     then element←(makeFileName ParameterElements:1)
                   else (AffirmError "Expecting an object class and a sequence of elements!"))
                 (if object←(DetermineObjectClass element objectClasses)
                   else (AffirmError <"Unknown object class:" element>)))
          (RETURN object])
```

32

```
(OpenAnnotation
  [LAMBDA (commandFile)                              (* R.Bates "11-Aug-80 09:21")
    (if (PRINTCOMMAND or commandFile=Terminal) and Annotating
        then AnnotationOpen←T])
```

33

```
(PerformAutoAppliedCommands
  [LAMBDA (command result abortCondition)                    (* D.Thompson "29-Oct-80 14:03")

        (* * This routine sees if the auto mechanism should be invoked after a command event. The command is used to
        categorize the event.)


    (PROG (oldPropn)
          (if abortCondition='finished
              then (if CurrentCommand MEMB CommandCategories:TheoremProverMachine
                       then [if CurrentTheorem
                                then (if CurrentCommand MEMB CommandCategories:ProofMaintenance
                                         then                  (* proof maintenance commands don't affect the current
                                                               proposition or the current theorem)
                                              (AutoMechanism 'ProofMaintenanceCommand)
                                       else                    (* a theorem prover command that has potentially changed
                                                               the current proposition and/or the status of some proof
                                                               tree)
                                            (SELECTQ CurrentCommand
                                                     (assume (AutoMechanism 'TheoremAssumed))
                                                     (apply

        (* An APPLY command: see if AutoSearch reduces the current proposition to TRUE, and the auto mechanism finds a new
        proposition. If so, print it out)

                                                                oldPropn←CurrentPropn
                                                                (AutoMechanism 'LemmaApplication)
                                                                (if oldPropn~=CurrentPropn
                                                                    then (PrintCurrentProposition T)))
                                                     (PROGN (if result
                                                                then
                                                                (* a command that has affected the current proposition.
                                                                result = T means don't invoke Auto Mechanism)
                                                                       (if result~=T
                                                                           then (AutoMechanism
                                                                                    'ProofCycleStep))
                                                                       (PrintCurrentProposition result]
                     elseif CurrentCommand MEMB CommandCategories:SpecificationMachine
                       then (AutoMechanism 'SpecificationCommandCompletion)
                     elseif CurrentCommand MEMB CommandCategories:ExecutiveMachine
                       then (AutoMechanism 'ExecutiveCommandCompletion)
                     elseif CurrentCommand MEMB CommandCategories:MiscellaneousMachine
                       then (AutoMechanism 'MiscellaneousCommandCompletion)
                     else (AutoMechanism 'CommandCompletion])
```

34

```
(PleaseDeclare
  [LAMBDA (what kind)                                        (* R.Erickson "23-Feb-81 20:47")
                                                             (* what is a new atom, kind is in
                                                             (Interface Variable Discard))

        (* * The user has used an unknown var or interface. Ask him/her to define. Return what if successful, ERROR! If
        refused.)


    (PRINTBELLS)
    (DOBE)
    (CLEARBUF T T)
    (printout NIL .TAB0 0)
    (SELECTQ kind
             (Interface (printout NIL "Please provide an interface declaration for" , (Shorten what)))
             (Variable (printout NIL "Please declare" , (Shorten what)))
             (Discard (printout NIL "To proceed, discard the variable"))
             (CannedMessage 'key <kind 'PleaseDeclare >))
    (printout NIL T "then type ok; (to continue) or abort; (to stop the parent command)")
    (if (Dtvs Terminal)='aborted
        then (ERROR!)
      else what])
```

35

```
(Plural
  [LAMBDA (object list)                                      (* D.Thompson "19-Sep-80 12:35")
```

```
(* * This routine takes a list length (or a list) and a noun (or string containing a noun and a verb) as its
parameters, and ensures agreement in number between the noun, the verb, and the list or length by modifying the
verb. -
NOTE: The nouns must be "pluralized" in the normal "add s" manner, and the verbs currently processed are "is" and
"are.")


(PROG (singularList verbPos)
      (singularList←(if (FIXP list)
                        then list=1
                      else (EQLENGTH (MKLIST list)
                                     1)))
      (RETURN (if (STRINGP object)
                  then (if verbPos←(VerbInString object 'is)
                           then                                  (* singular form)
                                (if singularList
                                    then object
                                  else (CONCAT (OR (SUBSTRING object 1 verbPos:1)
                                                   NullString)
                                               "s are"
                                               (OR (SUBSTRING object verbPos:2)
                                                   NullString)))
                         elseif verbPos←(VerbInString object 'are)
                           then                                  (* plural form)
                                (if singularList
                                    then (CONCAT (OR (SUBSTRING object 1 verbPos:1)
                                                     NullString)
                                                 " is"
                                                 (OR (SUBSTRING object verbPos:2)
                                                     NullString))
                                  else object)
                         else                                    (* no recognizable verb was found.)
                              object)
                elseif (LITATOM object)
                  then (if object MEMB '(are Are is Is)
                           then                                  (* a verb rather than a noun.)
                                (if singularList
                                    then (if (NTHCHAR object 1) MEMB '(A I)
                                             then 'Is
                                           else 'is)
                                  else (if (NTHCHAR object 1) MEMB '(A I)
                                           then 'Are
                                         else 'are))
                         elseif singularList
                           then (RemoveChar object 's)
                         else (AppendChar object 's))
                else                                             (* not an object we can deal with.)
                     object])
```

36

(PopTypeStack

```
[LAMBDA NIL                                                      (* D.Thompson " 7-Aug-80 17:58")

(* * This routine moves to the previously edited type on the type stack if possible.)


(if TypeStack::1
    then (printout NIL .TAB0 0 "Leaving" , CurrentType "; now editing" , TypeStack:2 Period T)
         (PROG1 CurrentType TypeStack←TypeStack::1
                CurrentType←TypeStack:1
                CurrentContext←CurrentType:LocalDeclarations)
  else NIL])
```

37

(PrintEnd

```
[LAMBDA (command commandFile)                                    (* D.Thompson "28-Aug-80 11:54")

(* * ensures we are positioned at the left margin. Currently this is only for input from the terminal.
Soon to use the Echoing flag.)


(if commandFile=Terminal and command ~MEMB LispCommands
    then (printout NIL .TAB0 0])
```

38

```
(PrintPrompt
  [LAMBDA (commandFile LowerExec)                    (* R.Bates " 5-Aug-80 16.05")

          (* * This routine prints the current event number and the prompt sequence. The routine also warns the user if
          there's no transcript file.)


    (PROG (separator modulus)
          (if commandFile=Terminal or PRINTCOMMAND
              then CurrentEventNumber←LISPXHISTORY:2+1      (* Code copied from lisp's PROMPTCHAR.
                                                            Compute the # of the upcoming event.)
                   (if CurrentEventNumber gt modulus←(LISPXHISTORY:4 or 100)
                       then CurrentEventNumber←CurrentEventNumber-modulus)
                   (if (DRIBBLEFILE)
                       then separator←Blank
                     else separator←Asterisk
                          (RecoverTranscript))
                   (if PromptPrinted
                       then (Dprintout T .TAB0 0 CurrentEventNumber)
                            (printName ExclamationPoint 'AnnotationFlag)
                            (if LowerExec gt 0
                                then (Dprintout NIL T "(" LowerExec ")" .SP LowerExec*INDENTATION))
                            (Dprintout T separator .PARA 0 0 (MKLIST Prompt)
                                       .)
                     else (printout NIL .TAB0 0 T CurrentEventNumber)
                          (printName ExclamationPoint 'AnnotationFlag)
                          (if LowerExec gt 0
                              then (printout NIL , "(" LowerExec ")" .SP (LowerExec*INDENTATION)))
                          (printout NIL separator # (printName Prompt 'ExecPrompt)
                                    .)))
          (PromptPrinted←NIL])
```



```
(ProcessCommand
  [LAMBDA (command parameters commandFile internalCall)      (* R.Erickson "23-Sep-81 17:17")

          (* * This routine is the central command routine. -
          Given a command and any necessary parameters, this routine first error-checks the parameters, and then invokes the
          command function that actually performs the command. A general escape for new commands is provided.)


          (* * This routine is expected to RETURN the value of the command function eventually called.)


    (SELECTQ command
             (; NIL)
             (@ (PROG (annotation editorCommands transformation)
                                                          (* In preparation for allowing the user to specify
                                                          editor commands as part of the command line)
                      (annotation←parameters)             (* (if editorCommands = NIL and commandFile~ = Terminal then
                                                          (AffirmError
"You didn't supply any editor commands, and you're reading from a file!")))
                      (transformation←(EditTheorem editorCommands))
                      (if ~annotation
                          then (printout NIL .TAB0 0 "Please summarize what you did, end with" ,
                                         SingleQuote CommandTerminator SingleQuote T)
                               annotation←(ReadRestOfLine commandFile T)
                                                          (* annotation is optional))
                      (if annotation
                          then (Annotate annotation transformation))
                                                          (* return an indication that some effect occurred.)
                      (RETURN CurrentPropn)))
             (abort (IgnoreExcess parameters)
                    (if (TopLevel?)
                        then (AffirmError "You're already at the top-level exec.")
                      else (RETFROM 'Dtvs 'aborted T)))
             [adopt (PROG (typeName)
                          (if typeName←(CheckForType parameters)
                              then (adopt typeName)
                            else (AffirmError]
             (affirmed? (IgnoreExcess parameters)
                        (affirmed?))
             [annotate (PROG (annotation node)
                             (if parameters
                                 then (if parameters:2=Comma
                                          then node←parameters:1
```

```
                                    annotation←parameters::2
                                else annotation←parameters)
                              (Annotate annotation (GetNode node))
                        else (AffirmError "You need to supply an annotation."]
(apply (use (nodeExpression (parse 'expressionSeq parameters CommandTerminator)
                                :expression)
            T))
(arc (if parameters
         then (IgnoreExcess parameters::1)
              (arc parameters:1)
       else (AffirmError "You need to supply an arc name.")))
(assume (Assume (nodeExpression (parse 'expressionSeq parameters CommandTerminator)
                                :expression)))
(augment (augment (ReadExpression parameters CommandTerminator)))
(automatic                                          (* will print message eventually)
           NIL)
[axiom (PROG (x)
             (if x←(parse 'ruleSeq parameters CommandTerminator)
                 then (ruleSeqParse x 'axiom)
              else (AffirmError]
(batch (if parameters
           then [batch (AFFIRMSpellingCorrect (L-CASE parameters:1)
                                              '(on off semi))
                       (fetch op of (if (parse 'interfaceList parameters::1
                                               CommandTerminator)
                                        else (AffirmError]
         else (batch)))
(cases (IgnoreExcess parameters)
       (cases))
[choose (PROG (y)
              (if y←(parse 'interfaceList parameters CommandTerminator):op
                   and (for x in y always (FIXP x) and x gt 0)
                 then (RETURN (choose y))
               else (AffirmError (if y
                                     then
                                   "All arguments to choose must be positive integers."
                                  else NIL]
(clear (IgnoreExcess parameters::1)
       (if (AFFIRMSpellingCorrect (L-CASE parameters:1)
                                  '(proof)
                                  NIL T T)
         then (ClearProof)
         else (AffirmError "Clear what? ('proof' is the only choice currently!)")))
(compile (PROG (object)
               (if parameters
                 else (AffirmError
                                "Expecting an object class and a sequence of objects!"))
               (object←(ObjectClassOfParameters parameters '(Types Files)
                                                internalCall))
               (AffirmCompile object ParameterElements)))
(complete (IgnoreExcess parameters)
          (complete))
[declare (PROG (x)
               (if x←(parse 'variableDecl parameters CommandTerminator)
                   then (declareVar x:identifier x:expression 'user)
                 else (AffirmError]
[define (PROG (x)
              (if x←(parse 'ruleSeq parameters CommandTerminator)
                  then (ruleSeqParse x 'define)
                 else (AffirmError]
[denote (PROG (x)
              (x←(parse 'denoteSpec parameters CommandTerminator))
              (if x
                  then (RETURN (denote (for pair in x:denotePair
                                           collect <pair:expression pair:identifier>)))
                 else (AffirmError]
[distinct (PROG (x)
                (if x←(parse 'distinctSpec parameters CommandTerminator)
                    then (distinct x:identifier)
                 else (AffirmError]
[discard (PROG (elements object)
               (object←(AFFIRMSpellingCorrect parameters:1 KnownNames:DiscardOptions
                                              QuestionMark NIL T))
               (if object=QuestionMark
                   then (printHelper 'known '(DiscardOptions))
                 else object←(AFFIRMSpellingCorrect object KnownNames:AFFIRMObjects NIL
                                                    T T)
                      elements←(SELECTQ object
                                        (Interfaces (parse 'interfaceList parameters::1
                                                           CommandTerminator)
                                                    :op)
```

```
                                        ((Lhs Nodes Theorems Variables)
                                          (parse 'expressionSeq parameters::1
                                                 CommandTerminator)
                                          :expression)
                                        (PROGN (IgnoreExcess parameters::1)
                                               NIL))
                          (discard object elements]
(down (if parameters
          then (IgnoreExcess parameters::1)
               (down parameters:1)
        else (down)))
(e (LISPX parameters))
[edit (PROG (x)
            (if x←(parse 'expression parameters CommandTerminator)
                then (Edit x)
              else (AffirmError]
(employ (employ (ReadExpression parameters CommandTerminator)))
(end (IgnoreExcess parameters)
     (AutoMechanism 'TypeClosed)
     (if (PopTypeStack)
       else (printout NIL .TAB0 0 "You can't close" , TypeStack:1 Period T)
            (AffirmError)))
(enter (enter (for x in parameters collect x unless x=Comma)))
[eval (PROG (x)
            (if x←(parse 'expression parameters CommandTerminator) and x←(pexec x)
                then (PrettyPrint (EVAL x))
              else (AffirmError]
(exec (IgnoreExcess parameters)
      (exec))
(fix (redo parameters commandFile T))
(forget (IgnoreExcess parameters)
        (printout NIL .TAB0 0 "Please use the 'discard history' command;" ,
                  "the 'forget' command will disappear soon."
                  T)
        (forget))
(freeze (PROG (file)
              (printout NIL .TAB0 0 "Writing file ..." , #
                        (if [LISTP file←(SYSOUT (makeFileName (if parameters
                                                                  else (UserProfile
                                                                       'FreezeFileName]
                            then ComingBackFromSYSOUT←T
                                 (if UsingTed
                                     then STOPAFFIRM←T
                                   else STOPAFFIRM←NIL)
                                 (if commandFile and commandFile~=Terminal
                                     then (RETFROM 'Dtvs 'aborted T)))
                        file T)
              (RETURN file)))
[genvcs (genvcs (if (parse 'interfaceList parameters CommandTerminator):op
                  else (AffirmError]
[gripe (PROG (keyList optionList question subject)
             (if parameters
                 then (if (NCHARS subject←(SUBSTRING parameters 2 -2)) gt
                          GripeSubjectMaxLength
                          then subject←(SUBSTRING subject 1 GripeSubjectMaxLength)
                               (printout NIL .TAB0 0
                                         "Your gripe subject must be truncated to"
                                         , GripeSubjectMaxLength ,
                                         "characters (the max):"
                                         T DoubleQuote subject DoubleQuote Period T)
                               question←"Do you want to retype it? " keyList←NIL
                               optionList←NIL
                               (if (AFFIRMUSER NIL 'N question keyList T NIL optionList)
                                   ='Y
                                   then (AffirmError)))
                      (gripe subject)
               else (AffirmError <"You need a subject (" GripeSubjectMaxLength
                                  "characters max)."
                                  >]
(help (AFFIRMHelp 'Exec parameters))
(ImplicitE (LISPX parameters))
(infix (Infix (if (parse 'interfaceList parameters CommandTerminator):op
                else (AffirmError))
              CurrentType))
[interface (PROG (x type)
                 (if x←(parse 'functionDecl parameters CommandTerminator)
                     then type←(CheckForType x:expression# NIL 'declare)
                          (for y in x:expression do (DeclareFun y type))
                          (if type~=CurrentType
                              then (AddNeeds 'Types type CurrentType T))
                   else (AffirmError]
```

```
(invoke (invoke (parse 'rangedInterfaceList (COPY parameters)
                       CommandTerminator)))
(keep (if (ILEQ (FLENGTH parameters←(for x in parameters collect x unless x=Comma))
                1)
          then (AffirmError
                 "The keep command needs a group name and a list of proposition names.")
        else (keep parameters:1 parameters::1)))
[lemma (printout NIL .TAB0 0 "Please use the 'rulelemma' command." T
                 "The 'lemma' command will disappear soon."
                 T)
       (PROG (x)
             (if x←(parse 'ruleSeq parameters CommandTerminator)
                 then (ruleSeqParse x 'lemma)
               else (AffirmError]
[let (PROG (x)
           (if x←(parse 'expressionSeq parameters CommandTerminator)
                 and x←[pexec (if (LENGTH x←x:expression)=1
                                  then x:1
                                else (PROG ((ANDOP ANDIO))
                                           (RETURN (N2BINARY <ANDIO ! x>]
               then (RETURN (let x NIL))
             else (AffirmError]
(lisp (IgnoreExcess parameters)
      (printout NIL .TAB0 0 "Type OK to return to AFFIRM." T)
      (LowerInterLisp))
(load (PROG (fileName method)
            (fileName←parameters)
            (if parameters:1 and method←(AFFIRMSpellingCorrect (L-CASE parameters:1)
                                                               '(fast slow)
                                                               NIL T NIL)
                then fileName←parameters::1)
            (AffirmLoad fileName method)))
(monitor (monitor parameters))
[name (PROG (x)
            (if x←(parse 'expressionSeq parameters CommandTerminator)
                then (name x:expression)
              else (AffirmError]
[needs (PROG (object)
             (if parameters
                 then object←(ObjectClassOfParameters parameters '(Types Files)
                                                      internalCall)
                      (needs object ParameterElements CurrentType)
               else (AffirmError "Expecting an object class and a sequence of objects!"]
(next (IgnoreExcess parameters)
      (next))
[nochange (PROG (x)
                (x←(parse 'nochangeSpec parameters CommandTerminator))
                (if x
                    then (nochange x:identifier x:expression)
                  else (AffirmError]
(normalize (IgnoreExcess parameters)
           (Normalize T T))
(normint (IgnoreExcess parameters)
         (normint))
(note NIL)
(ok (IgnoreExcess parameters)
    (if (TopLevel?)
        then (AffirmError "You're already at the top-level exec.")
      else (RETFROM 'Dtvs T T)))
(pause (IgnoreExcess parameters)
       (ReadCommandFile Terminal))
[print (RESETFORM (GCGAG NIL)
                  (print (for x in parameters collect x unless x=Comma]
(profile (profile parameters))
[put (PROG (x)
           (if x←(parse 'expressionSeq parameters CommandTerminator)
                 and x←[pexec (if (LENGTH x←x:expression)=1
                                  then x:1
                                else (PROG ((ANDOP ANDIO))
                                           (RETURN (N2BINARY <ANDIO ! x>]
               then (RETURN (let x T))
             else (AffirmError]
(quit (if (UserProfile 'AutoFreeze T)
          else (IgnoreExcess parameters))
      ComingBackFromSYSOUT←NIL
      (AutoMechanism 'SessionCompletion parameters)
      (if ComingBackFromSYSOUT
          else (printout NIL .TAB0 0 "Type CONTINUE to return to AFFIRM." T)
               (LOGOUT)))
(read (DTVS parameters))
(readp (readp parameters))
```

```
(redo (redo parameters commandFile NIL))
(renumber (renumber parameters))
[replace (replaceCommand (if parameters
                             then (ReadExpressionList parameters CommandTerminator]
(resume (IgnoreExcess parameters)
        (resume))
(retry (IgnoreExcess parameters)
       (retry))
(review (IgnoreExcess parameters)
        (CloseDribble))
[rulelemma (PROG (x)
                 (if x←(parse 'ruleSeq parameters CommandTerminator)
                     then (ruleSeqParse x 'lemma)
                   else (AffirmError]
(save (PROG (method object)
            (if parameters
              else (AffirmError "Expecting an object class and a sequence of objects!"))
            (if method←(AFFIRMSpellingCorrect (L-CASE parameters:1)
                                              '(fast slow)
                                              NIL T NIL)
                then (pop parameters)
                     (if parameters
                       else (AffirmError
                              "Expecting an object class and a sequence of objects!")))
            (object←(ObjectClassOfParameters parameters '(Types Files)
                                             internalCall))
            (save object ParameterElements method)))
[schema (PROG (x)
              (if x←(parse 'ruleSeq parameters CommandTerminator)
                  then (ruleSeqParse x 'schema)
                else (AffirmError]
(search (IgnoreExcess parameters)
        (search))
[set (PROG (x y parameters1 parameters2)
           (parameters2←(('TO MEMB parameters)::1 or ('to MEMB parameters)::1))
           (parameters1←(LDIFF parameters parameters2))
           (x←(parse 'expression parameters1 'TO))
           (y←(parse 'expression parameters2 CommandTerminator))
           (if x and y and x←(pexec x)
               then (RETURN (set x y))
             else (AffirmError]
(split (IgnoreExcess parameters)
       (split))
(stop (IgnoreExcess parameters)
      (if (TopLevel?)
          then (AffirmError "You're already at the top-level exec.")
        else (CLOSEF? (INFILE oldinputfile))
             (RETFROM 'DTVS T T)))
(storage (IgnoreExcess parameters::1)
         (storage parameters:1))
[(sufficient sufficient?)
  (PROG (typeName)
        (if typeName←(CheckForType parameters)
            then (SufficientlyComplete typeName]
(swap (swapCommand (parse 'rangedInterfaceList (COPY parameters)
                          CommandTerminator)
                   parameters))
(suppose (if parameters
             then (suppose (ReadExpression parameters CommandTerminator))
           else (split)))
[thaw (PROG (file temp)
            (if file←(getFileName <'BODY (if parameters
                                             then (PACK parameters)
                                           else (UserProfile 'FreezeFileName))
                                  'EXTENSION
                                  (if SYSTEMTYPE='TOPS20
                                      then 'EXE
                                    else 'SAV)
                                  >
                                  NIL "file to thaw: ")
                then (PROG (HELPFLAG)
                           (temp←(NLSETQ (SYSOUTP file)))
                           (if temp=NIL or ~(EQUAL temp '(NIL))
                               then (ERSETQ (SYSIN file)))
                           (AffirmError (if (EQUAL temp '(NIL))
                                            then <file "isn't a freeze file." >
                                          else NIL]
(theorem (theorem (nodeExpression (parse 'expressionSeq parameters CommandTerminator)
                                  :expression)))
(transcript (transcript parameters NIL))
(try (Try (nodeExpression (parse 'expressionSeq parameters CommandTerminator):expression)
```

```
                    ))
            [type (PROG (x)
                        (if x←(parse 'expression parameters CommandTerminator)
                            then (DeclareType x)
                                 (Edit x)
                                 (if BootStrapping=NIL
                                     then (GenerateReflexiveRule))
                          else (AffirmError]
            (undo (UNDOLISPX parameters))
            (up (if parameters
                    then (IgnoreExcess parameters::1)
                         (if (FIXP parameters:1) and parameters:1 gt 0
                             then (up parameters:1)
                          else (AffirmError "Expecting a positive integer!"))
                else (up 1)))
            (use (use (nodeExpression (parse 'expressionSeq parameters CommandTerminator):expression)
                      NIL))
            (PROGN (if (DefinedAFFIRMCommand command) and (FGETD command)
                       then (APPLY command (if command MEMB OwnReadCommands
                                               then <commandFile CommandTerminator>
                                            else parameters))
                  else (printout NIL .TAB0 0 (L-CASE command T)
                                 , "isn't currently defined as a command! Skipping to %""
                                 CommandTerminator "%" "..." ,)
                       (ReadRestOfLine commandFile T)
                       NIL])
```

40

(ProcessCommandAbort

```
  [LAMBDA (command parameters commandFile abortCondition)    (* D.Thompson " 7-Aug-80 13:17")
    (CloseAnnotation)
    (RestoreTerminalEnvironment)
    (SELECTQ abortCondition
            (T (printout NIL .TAB0 0 "(..." # (PRINTBELLS)
                         , command , "aborted: effects undone)" T)
               (if commandFile ~= Terminal and ~STOPAFFIRM
                   then (ErrorInFile command parameters)
                 elseif command~='fix
                   then AutoFix←NIL
                        (AutoMechanism 'CompletedCommandAbort)))
            (input (printout NIL .TAB0 0 "(..." # (PRINTBELLS)
                             , "command input aborted)" T))
            ((finished processing)
              (printout NIL .TAB0 0 # (PRINTBELLS)
                        (L-CASE command T)
                        , "was aborted during the undo-ing of its effects;" T
                        "undo-ing may not have completed."
                        T "Try an explicit UNDO command." T)
              (if commandFile ~= Terminal and ~STOPAFFIRM
                  then (ErrorInFile command parameters)
                elseif command~='fix
                  then AutoFix←NIL
                       (AutoMechanism 'IncompleteCommandAbort)))
            (Unexpected 'abortCondition])
```

41

(ReadCommandFile

```
  [LAMBDA (commandFile)                                    (* D.Thompson "10-Sep-80 12:54")
    (ProcessCommand 'read commandFile Terminal 'internal])
```

42

(readParams

```
  [LAMBDA (stop file readTable)                            (* D.Thompson "14-Jan-81 16:44")

          (* * This routine is kinda like RATOMS, except it handles strings as well.)


    (bind (c paramLine) until c←(RATOM file readTable)=stop finally (RETURN (ENDCOLLECT paramLine))
       do paramLine←(DOCOLLECT (if c=DoubleQuote
                                   then (readString file)
                                else c)
                               paramLine])
```

43

```
(ReadRestOfLine
  [LAMBDA (commandFile readTable)                    (* D.Thompson " 2-Sep-80 12:34")

          (* * This routine reads atoms from the current command file until the command terminator is found.
          -
          This routine is currently the main method of reading a command function's parameters.)


    (if readTable=T
        then readTable←PascalReadTable)
    (EatComments (if commandFile=Terminal
                     then (bind atom eachtime (if (READP Terminal)
                                                  else (Dprintout T PARSERPROMPT))
                                              (atom←(RATOM Terminal readTable))
                          until atom=CommandTerminator collect atom)
                 else (RATOMS CommandTerminator commandFile readTable])
```

44

```
(ReadSpecialCommandParameters
  [LAMBDA (command commandFile)                      (* D.Thompson " 5-Sep-80 14:15")

          (* * This routine reads the parameters for those command functions requiring some sort of special treatment.)


    (SELECTQ command
            (@ (if Annotating and commandFile=Terminal
                   then                              (* Scribe treats At specially, so we need a AtAt in
                                                     transcript. If buffering is off, we'll have the sequence
                                                     "At; C/R AtAt;")
                        (if (UserProfile 'CommandLineBuffering T)
                            then (Dprintout NIL AtSign AtSign SemiColon)
                          else                       (* immediately follows the user's @)
                               (Dprintout NIL AtSign)))
               (ReadRestOfLine commandFile T))
            (e (LISPXREAD commandFile))
            ((fix redo undo)
              (RESETLST (RESETSAVE (SETBRK <CommandTerminator LeftCurlyBracket RightCurlyBracket> 1)
                                   <'SETBRK (GETBRK)
                                    >)
                        (ReadRestOfLine commandFile NIL)))
            (ImplicitE (if OriginalCommandWord
                         else (LISPXREAD commandFile)))
            (note (RESETFORM (GCGAG NIL)
                             (ReadRestOfLine commandFile T)))
            (profile (EatComments (readParams CommandTerminator commandFile ProfileReadTable)))
            (stop (if commandFile=Terminal
                      then (ReadRestOfLine Terminal T)
                    else NIL))
            (Unexpected 'SpecialReadCommand])
```

45

```
(readString
  [LAMBDA (file)                                     (* D.Thompson "19-Aug-80 11:23")

          (* * This routine gathers the contents of a string from the primary input file. The first double quote has ALREADY
          been read; this routine reads up to and including the second (and handles embedded ones!).)


    (PROG (c done param)
          (until done finally param←(MKSTRING (PACK (ENDCOLLECT param)))
             do (c←(READC file))
                (if c=DoubleQuote
                    then done←T
                  else param←(DOCOLLECT (if c=PercentSign
                                            then (READC file)
                                          else c)
                                        param)))
          (RETURN param])
```

46

```
(RemoveChar
  [LAMBDA (atom character)                           (* D.Thompson "25-Nov-79 01:08")

          (* * This routine removes the character provided as the second parameter from the end of the atom provided as the
```

first parameter, if the character ends the atom.)

```
    (if (NTHCHAR atom -1)=character
        then (MKATOM (SUBSTRING atom 1 -2))
      else atom])
```

47

(RestoreTerminalEnvironment

```
  [LAMBDA NIL                                                    (* D.Thompson "19-Dec-79 12:33")
    (if CurrentTerminalEnvironment ~= NIL
        and (UNDONLSETQ (PROGN (CONTROL CurrentTerminalEnvironment:LineBuffering)
                               (ECHOMODE CurrentTerminalEnvironment:EchoMode)
                               (for value in CurrentTerminalEnvironment:EchoControls as i
                                  from 1 do (ECHOCONTROL i value))
                               (for type in DeleteControlTypes as value
                                  in CurrentTerminalEnvironment:DeleteControls
                                  do (DELETECONTROL type value))
                               (DELETECONTROL CurrentTerminalEnvironment:DeleteEchoMode)))
      then T
      else NIL])
```

48

(SaveTerminalEnvironment

```
  [LAMBDA NIL                                                    (* D.Thompson "12-Sep-79 17:53")
    (CurrentTerminalEnvironment←(create TerminalEnvironment LineBuffering←(GETCONTROL)
                                        EchoMode←(GETECHOMODE)
                                        EchoControls←(for i from 1 to 26 collect (ECHOCONTROL i))
                                        DeleteControls←(for type in DeleteControlTypes
                                                          collect (DELETECONTROL type))
                                        DeleteEchoMode←(GETDELETECONTROL 'ECHO])
```

49

(SetEventCompletionFlag

```
  [LAMBDA (command commandFile)                                  (* D.Thompson " 5-Sep-80 14:41")
    (if LISPXHISTORY and (LISTP LISPXHIST) and (NTH LISPXHIST 3)
        then (if commandFile=Terminal
                 then (if command MEMB LispCommands
                        else (LISPXSTOREVALUE LISPXHIST T])
```

50

(Singularize

```
  [LAMBDA (atom)
    (PROG (chars)
          (chars←(DREVERSE (UNPACK atom)))
          (RETURN (if chars:1='s
                      then (PACK (DREVERSE chars::1))
                    else atom])
```

51

(SkipOverComments

```
  [LAMBDA (commandFile)                                          (* D.Thompson " 2-Sep-80 11:30")
    (eachtime (SkipToRealChars commandFile) while (PEEKC commandFile)=LeftCurlyBracket
       do (skipPascalComment commandFile])
```

52

(SkipToRealChars

```
  [LAMBDA (commandFile)                                          (* D.Thompson "25-Sep-79 13:24")
    (while (PEEKC commandFile) MEMB <Blank CR TabChar> do (READC commandFile])
```

53

(StartMonitorCycle

```
  [LAMBDA NIL                                                    (* D.Thompson "25-Sep-79 13:19")
    (CLOCK 2])
```

54

```
(SuppressAdvisedPrint?
  [LAMBDA (whence)                                    (* R.Erickson " 2-Feb-81 20:02")

          (* * We are in an advice on PRINT. Return T if PRINT should not occur. Used to prevent the compiler messages
          "(dwimifying f'" and "(f (x) (integer))." -
          whence is the calling function. Sample advice: (OR (AND (FGETD (QUOTE SuppressAdvisedPrint?))
          (SuppressAdvisedPrint? (QUOTE LAP))) (PRINT --)))



          (* the global SuppressCompilerMsgs is one of -
          NIL no suppression -
          T suppress; compute and reset from the name of the type -
          a list of possible freevars. Any not in this list will cause the message to proceed.)


    (AND SuppressCompilerMsgs (SELECTQ whence
                                       (COMPILE1A                  (* always)
                                                  T)
                                       (LAP                        (* use PRINT's parameters, X FILE and RDTBL, to see if
                                                                   the right place?)
                                            (if X and FILE=COUTFILE and ~RDTBL
                                                then                (* X is a list (fn args trees); are all the trees okay?)
                                                     (if (NLISTP SuppressCompilerMsgs)
                                                         then
                                                         (* we need to compute. Fetch from "needs" of the
                                                         typename, since only type names may legitimately be used
                                                         freely.)
                                                           (PROG (ext need)
                                                                 (if ext←(Extension X:1)
                                                                     then need←(ASSOC 'Types
                                                                                      ext:Needs)::1
                                                                         SuppressCompilerMsgs←
                                                                          <ext ! need  :
                                                                            !
                                                                          PredefinedNames:Types>
                                                         (* :Needs doesn't include self, predefined)

                                                                             else
                                                         (* can't compute. avoid doing this over.)
                                                                                  SuppressCompilerMsgs← <T>))
                                                     )
                                                         (* all freevars kosher?)
                                                    ~(LDIFFERENCE X:3 SuppressCompilerMsgs)))
                                       (SHOULDNT])
```

55

```
(UpdateLineDeleteMessage
  [LAMBDA (switch)                                    (* D.Thompson " 6-Feb-80 13:19")

          (* * This routine updates the message LISP prints out when the user types control-Q (TENEX) or control-U
          (TOPS-20). AFFIRM users will see different (QUOTE deleted') confirmation messages depending on the current context:
          reading a command name; reading parameters; LISP executive; etc.)


    (DELETECONTROL 'LINEDELETE (SELECTQ (U-CASE switch)
                                        (COMMAND LineDeleteMessage:CommandEnv)
                                        (PARAMETERS LineDeleteMessage:ParameterEnv)
                                        (LISP LineDeleteMessage: NormalEnv)
                                        (PROGN (Unexpected 'LineDelete)
                                               LineDeleteMessage:NormalEnv])
```

56

```
(ValidateFileName
  [LAMBDA (proposedFileName objectCategory dontReturn)        (* D.Thompson " 8-Feb-80 09:45")

          (* * This routine checks its parameter to see if it can be a file name. -
          good for atomic names only! (not parameterized types))


    (PROG (nameOK)
          (nameOK←T)
          (if 'NOBIND ~=(EVALV (FILECOMS (U-CASE proposedFileName)))
              then (printout NIL .TAB0 0 "Sorry, the name" , proposedFileName ,
                             "would conflict with an existing LISP file name."
                             T)
```

```
                (if dontReturn
                    then (AffirmError)
                  else nameOK←NIL))
          (SELECTQ objectCategory
                (Types (if 'NOBIND ~=(EVALV proposedFileName)
                           or (STRPOS TypeSeparator proposedFileName)
                           or (STRPOS Period propos(dFileName) or proposedFileName MEMB
                                                                ReservedNames:Types
                           or (U-CASE proposedFileName)=NIL
                       then (printout NIL .TAB0 0 "Sorry, the name" , proposedFileName ,
                                      "is forbidden as a type name."
                                      T)
                            (if dontReturn
                                then (AffirmError)
                              else nameOK←NIL)))
                (PROGN (Unexpected 'AFFIRMObject)
                       nameOK←NIL))
          (RETURN nameOK])
```

57

```
(VerbInString
  [LAMBDA (string verb)                                    (* D.Thompson "23-Nov-79 13:14")

          (* * This routine finds the character position in its subject string of a verb provided as the second parameter.
          Currently the recognized verbs are "is" and "are".)


    (PROG (first)
          (RETURN (if (STRINGP string)
                      then (SELECTQ verb
                                    (is (if first←(STRPOS " is" string)
                                            then <first-1 first+3>))
                                    (are (if first←(STRPOS "s are" string)
                                             then <first-1 first+5>
                                           elseif first←(STRPOS " are" string)
                                             then <first-1 first+4>))
                                    (Unexpected 'StringVerb])
)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* Command Routines) ]

(DEFINEQ
```

58

```
(AffirmCompile
  [LAMBDA (object fileNames)                                    (* D.Thompson "10-Sep-80 12:50")

          (* * This routine oversees the compilation of a file.)


    (UPDATEFILES)
    (PROG (SuppressCompilerMsgs fileName)
          (for f in fileNames
             do (fileName←(FILENAMEFIELD (makeFileName f)
                                         'NAME))
                (if (GETPROP fileName 'FILE)::1
                    then                                        (* changes since last saved)
                         (printout NIL .TAB0 0 "The current version of" , fileName ,
                                   "hasn't yet been saved."
                                   , "Saving it now." T)
                         (ProcessCommand 'save <object f> Terminal 'internal))
                (LISPXUNREAD (if (UserProfile 'CompileOption)='Redefine
                                 then (if (Loaded? (INFILEP fileName)
                                                   'Files)
                                          then '(STF)
                                        else (AffirmError <"Top version of " fileName
                                                          "is not loaded, compiling to file only"
                                                            >
                                                          'mild)
                                             '(F))
                                 else '(F)))
                (SuppressCompilerMsgs←T)                        (* don't give routine messages;
                                                                   will compute permissible freevars from Needs)
                (BCOMPL fileName (if object~='Files
                                     then (getFileNameFromDirectory fileName NIL object 'OldCompiled
                                                                    'OUTPUT])
```

59

```
(AffirmLoad
  [LAMBDA (fileName method)                                     (* D.Thompson " 5-Sep-80 19:56")

          (* * This routine loads the file provided as the first parameter.)


    method←(if method
               else (L-CASE (UserProfile 'LoadMethod)))
    (if fileName←(getFileName fileName NIL "File to load: ")
        then [printout NIL .TAB0 0 (if method='slow
                                       then (LOAD fileName)
                                     else (PROG (ADDSPELLFLG BUILDMAPFLG)
                                                (RETURN (LOAD fileName T]
      else (AffirmError])
```

60

```
(AffirmNews
  [LAMBDA (FileName Open)                                       (* R.Bates " 6-Aug-80 10:28")
    (PROG (pos oldNewsDate)
          (if ~Open
              then (PROG (HELPFLAG)
                         (OR (ERSETQ FileName←(OPENFILE FileName 'BOTH))
                             (ERSETQ (PROGN Open←'ERROR
                                            FileName←(OPENFILE FileName 'INPUT)))
                             FileName←NIL)))
          (if FileName=NIL
              then (RETURN 'ABORTED))
          (FILEPOS 'STOP FileName)
          (pos←(SETFILEPTR FileName (GETFILEPTR FileName)+ 6))
          (oldNewsDate←(if (GETFILEPTR FileName)=(GETEOFPTR FileName)
                           then 0
                         else (RATOM FileName)))
          (SETFILEPTR FileName pos)
          (if Open~='ERROR
              then (printout FileName (IDATE)
                             T))
```

```
        (PrintNews oldNewsDate)
        (if ~Open
            then (CLOSEF FileName])
```

61

```
(AffirmSave
  [LAMBDA (name object method)                          (* R.Erickson "17-Mar-81 19:36")

        (* * This routine writes a LISP-format file. The name of the file depends on the object class of the object being
        written; file naming conventions are followed.)


    (PROG (fileName options)
          [method←(L-CASE (if method
                              else (UserProfile 'SaveMethod]
          (fileName←(getFileNameFromDirectory name NIL object 'OldSaved 'OUTPUT))
                                                        (* full file name)
          (if method='slow
              then options←'(CLISPIFY)
            else options←'(FAST NOCLISP))               (* LISP's MAKEFILE options)
          (if (GETPROP (FILENAMEFIELD fileName 'NAME)
                       'FILEDATES)
            else                                        (* no earlier loaded versions -- avoid message)
                 (push options 'NEW)))
          (printout NIL .TAB0 0 "Writing file..." , [if method='slow
                                                        then (PROG (HELPFLAG)
                                                                   (RETURN (MAKEFILE fileName options)
                                                                           ))
                                                      else (PROG (FONTCHANGEFLG CLISPIFYPRETTYFLG
                                                                                HELPFLAG
                                                                                (SAVEDBFLG 'NO)))
                                                           (RETURN (MAKEFILE fileName options]
                    T])
```

62

```
(batch
  [LAMBDA (switch answerList)                           (* D.Thompson "30-Jan-80 08:23")

        (* * This routine turns on and off the batch mode. -
        An AFFIRM command function.)


    (if switch
      else switch←'on)
    (SELECTQ (AFFIRMSpellingCorrect (L-CASE switch)
                                    '(off on semi))
             (on BatchMode←T BatchAnswers← (MKLIST answerList)
                 OldBreakAccess←
                 (UserProfile 'BreakAccess)
                 (UserProfileSet 'BreakAccess 'Off)     (* set terminal to no paging?)
                 )
             (off (if BatchMode and OldBreakAccess
                      then (UserProfileSet 'BreakAccess 'On))
                  BatchMode←NIL BatchAnswers←NIL)
             (semi BatchMode←'semi BatchAnswers← (MKLIST answerList)
                   OldBreakAccess←
                   (UserProfile 'BreakAccess)
                   (UserProfileSet 'BreakAccess 'Off))
             (AffirmError))
    BatchMode])
```

63

```
(EstablishInterLispEnvironment
  [LAMBDA NIL                                           (* D.Thompson "17-Oct-79 14:45")
    (UpdateLineDeleteMessage 'Lisp)
    (if (UserProfile 'ActivateInterLispEnvironment T)
        then (PROG1 (create AFFIRMEnvironment
                            BreakAccess ←(UserProfile 'BreakAccess)
                            LineLength ←(UserProfile 'TerminalLineWidth)
                            GCMessage ←(UserProfile 'GarbageCollectionMessage)
                            GCPages ←(UserProfile 'GarbageCollectionPages)
                            HistoryLength ←(UserProfile 'HistoryWindowSize))
                    (UserProfileSet 'BreakAccess (UserProfile 'LISPBreakAccess))
                    (UserProfileSet 'TerminalLineWidth (UserProfile 'LISPTerminalLineWidth))
                    (UserProfileSet 'GarbageCollectionMessage (UserProfile
```

```
                                                        'LISPGarbageCollectionMessage))
                (UserProfileSet 'GarbageCollectionPages (UserProfile 'LISPGarbageCollectionPages))
                (UserProfileSet 'HistoryWindowSize (UserProfile 'LISPHistoryWindowSize])
```

64

```
(gripe
  [LAMBDA (subject)                                    (* R.Erickson " 3-Feb-81 19:41")

        (* * This routine accepts text from the user and send the text as a message to the PV group.
        The parameter is the subject of the message. -
        An AFFIRM command function.)


    (PROG (dribbleFile error keyList messageCopy optionList pvReportMail question StartBlockE
                  tempFile)
          (subject←(CONCAT subject Blank (if (EVALV 'AFFIRMVERSION)='NOBIND
                                             then "??"
                                           else (SUBSTRING AFFIRMVERSION 1 3) or AFFIRMVERSION)))
          (subject←((SUBSTRING subject 1 40) or subject))
          (tempFile←(OPENFILE 'SCRATCH.AFFIRM 'OUTPUT))
          (dribbleFile←(DRIBBLEFILE))
          (EvalFormWithOutDribble error←(CallSubsys 'SNDMSG 4 '((2 21474836480))
                                                    tempFile subject)
                                 (if error=NIL
                                     then (AffirmError
                                                "The attempt to send the gripe message failed."))
                                 (if dribbleFile
                                       and messageCopy←(INFILEP (PACKFILENAME 'DIRECTORY
                                                                              (DIRECTORYNAME)
                                                                              'NAME
                                                                              (if SYSTEMTYPE='TOPS20
                                                                                  then 'MAIL
                                                                                else 'MESSAGE)
                                                                              'EXTENSION 'COPY))
                                     then (OPENFILE messageCopy 'INPUT)
                                          (OPENFILE dribbleFile 'APPEND)
                                          (COPYBYTES messageCopy dribbleFile)
                                          (CLOSEF messageCopy)
                                          (CLOSEF dribbleFile)))
          (keyList←'((Abort NIL)
             (Sndmsg NIL)))
          (question←"Do you want to Sndmsg, or Abort? ")
          (optionList←'(AUTOCOMPLETEFLG T))
          [if (AFFIRMUSER NIL 'Sndmsg question keyList T NIL optionList)='Sndmsg
              then (for (user userMail) in PVMAINTAINERS first tempFile←(OPENFILE 'SCRATCH.AFFIRM
                                                                                  'INPUT)
                        do (userMail←(OPENFILE (PACKFILENAME 'DIRECTORY (DIRECTORYNAME)
                                                             'NAME "↑V[--UNSENT-MAIL--↑V]" 'EXTENSION
                                                             (CONCAT user "↑V@" HOMEMACHINE))
                                               'OUTPUT))
                           (COPYBYTES tempFile userMail)
                           (SETFILEPTR tempFile 0)
                           (CLOSEF userMail)
                        finally (pvReportMail←(OPENFILE (PACKFILENAME 'DIRECTORY (DIRECTORYNAME)
                                                                      'NAME "↑V[--UNSENT-MAIL--↑V]"
                                                                      'EXTENSION
                                                                      (CONCAT "PVREPORT" "↑V@"
                                                                              HOMEMACHINE))
                                                        'OUTPUT))
                                (JSYS 18 (LOGOR -34359738368 (OPNJFN pvReportMail)))
                                (JSYS 18 (LOGOR -34359738368 (OPNJFN tempFile)))
                                (RENAMEFILE tempFile pvReportMail)
                                (CLOSEF? pvReportMail)
                                (RLJFN -1)
                                (question←"Do you want to send along the transcript? ")
                                (keyList←NIL)
                                (optionList←NIL)
                                (if dribbleFile←(DRIBBLEFILE)
                                      and (AFFIRMUSER NIL 'N question keyList T NIL optionList)='Y
                                    then pvReportMail←(OPENFILE (PACKFILENAME 'DIRECTORY (DIRECTORYNAME)
                                                                              'NAME
                                                                              "↑V[--UNSENT-MAIL--↑V]"
                                                                              'EXTENSION
                                                                              (CONCAT "PVREPORT" "↑V@"
                                                                                      HOMEMACHINE))
                                                            'OUTPUT)
                                         (printout pvReportMail "Date:" (DATE)
                                                   T "From: " USERNAME " at " (HOSTNAME)
                                                   T "Subject: transcript " subject T
```

```
                                        "To:  PVREPORT at "
                                        HOMEMACHINE T T)
                                    (EvalFormWithOutDribble (OPENFILE dribbleFile 'INPUT)
                                                            (COPYBYTES dribbleFile pvReportMail)
                                                            (CLOSEF dribbleFile))
                                    (printout pvReportMail "-------" T)
                                    (CLOSEF pvReportMail)))
                (printout NIL .TAB0 0 "The gripe message has been queued." T)
                (NotifyMailer (LOGAND (USERNUMBER)
                                      (CONSTANT (MKATOM "777777Q"]
        (DELFILE tempFile)
        (CLOSEF? tempFile)
        (RETURN tempFile])
```

65

```
(limboCommand
  [LAMBDA (command)                                    (* D.Thompson " 6-Nov-79 17:58")
    (printout NIL .TAB0 0 "Whoops!" T "Sorry, but this command doesn't currently do anything." T
              "We didn't realize anybody was still using it."
              T "Send a message to DTHOMPSON@ISIC, and we'll get it back in for you." T)
    (AffirmError])
```

66

```
(LowerInterLisp
  [LAMBDA NIL                                          (* D.Thompson " 4-Oct-79 16:31")
    (PROG ((ADDSPELLFLG T)
           currentAFFIRMEnvironment)
          (currentAFFIRMEnvironment←(EstablishInterLispEnvironment))
          (USEREXEC)
          (RestoreAFFIRMEnvironment currentAFFIRMEnvironment])
```

67

```
(monitor
  [LAMBDA (param)                                      (* D.Thompson " 5-Nov-79 15:10")

          (* * DHT: This is a dummy routine until I get this implemented.)


    (printout NIL .TAB0 0 "This capability is not yet functional." T])
```

68

```
(redo
  [LAMBDA (line commandFile fixCommand)                (* D.Thompson "10-Sep-80 12:56")

          (* * This routine performs the AFFIRM version of LISP's REDO command. -
          An AFFIRM command function.)


    (PROG (command event eventInput eventPrompt lastAtom parameters)
          (if (NLSETQ event←(LISPXFIND LISPXHISTORY line 'ENTRY T))
              then eventInput←event:1
                   eventPrompt←event:2
                   (if eventInput~=NIL
                       then (if (MEMBER eventPrompt <NormalPrompt BatchErrorPrompt>)
                                then                                  (* this is an AFFIRM command)
                                     (if eventInput:1 MEMB '(fix redo)
                                         then eventInput←eventInput:-1)
                                     (if fixCommand
                                         then eventInput←(MKLIST (UserEdit (COPY eventInput)
                                                                           PascalReadTable))
                                              command←(AFFIRMSpellingCorrect (L-CASE eventInput:1)
                                                                             KnownNames:Commands)
                                              parameters←eventInput::1
                                       else                           (* this is a REDO command)
                                            command←eventInput:1
                                            parameters←(COPY eventInput::1)
                                            (printout NIL .TAB0 0 "Redoing" , #
                                                      (PrettyPrintCommand <command ! parameters> NIL
                                                                          PascalReadTable)))
                                     (AddParametersToHistory 'redo commandFile
                                                             <<command ! parameters>>)
                                     CurrentCommand←command
                                     (if CommandTerminator=(CAR (lastAtom←(LAST parameters)))
```

```
                                    then parameters←(LDIFF parameters lastAtom))
                                  (RETURN (ProcessCommand command parameters Terminal NIL))
                           else                                    (* this is a LISP command)
                                (if (ATOM eventInput)
                                    then (LISPX eventInput)
                                  else (LISPXUNREAD eventInput::1)
                                       (LISPX eventInput:1)))
                      else (printout NIL .TAB0 0 "No information was saved by that event." T))
                 else (AffirmError <line "not found." >])
```

69

```
(renumber
  [LAMBDA (eventNumber)                                    (* D.Thompson " 3-Mar-80 18:40")

          (* * This routine resets the event number counter to be the parameter. -
          An AFFIRM command function.)


    (if (LISTP eventNumber)
        then eventNumber←(PACK eventNumber))
    (if eventNumber=NIL
        then eventNumber←1)
    (if (FIXP eventNumber)
        then (if eventNumber gt 0 and (ILEQ eventNumber LISPXHISTORY:4)
                 then (LISPXHISTORY:2←eventNumber-1)
               elseif (MINUSP eventNumber) and (ABS eventNumber) lt LISPXHISTORY:4
                 then (LISPXHISTORY:2←LISPXHISTORY:4+eventNumber)
               else (printout NIL .TAB0 0 "The new event number must be an integer in the range [1,"
                                  , LISPXHISTORY:4 "]." T))
      else (printout NIL .TAB0 0 "Expecting an integer!"))
    LISPXHISTORY:2+1])
```

70

```
(RestoreAFFIRMEnvironment
  [LAMBDA (savedParameters)                                (* D.Thompson "12-Sep-79 11:18")
    (if savedParameters ~= NIL
        then (UserProfileSet 'BreakAccess savedParameters:BreakAccess)
             (UserProfileSet 'TerminalLineWidth savedParameters:LineLength)
             (UserProfileSet 'GarbageCollectionMessage savedParameters:GCMessage)
             (UserProfileSet 'GarbageCollectionPages savedParameters:GCPages)
             (UserProfileSet 'HistoryWindowSize savedParameters:HistoryLength)
             T
      else NIL])
```

71

```
(save
  [LAMBDA (object elements method)                         (* D.Thompson " 8-Feb-80 10:40")

          (* * This routine oversees the saving of objects (such as types or files).)


    elements←(MKLIST elements)
    (SELECTQ object
            [Types (for type in elements bind n name
                      do (n←(if (LISTP type)
                                then (PACK type)
                              else type))
                         (if name←(CheckForType n NIL 'OK)
                             then (AffirmSave name 'Types method)
                           else (printout NIL .TAB0 0 n QuestionMark T]
            [Files (for file in elements bind name
                      do (name←(makeFileName file))
                         (if (Loaded? name 'Files)
                             then (AffirmSave name 'Files method)
                           else (printout NIL .TAB0 0 name QuestionMark T]
            (Unexpected 'AFFIRMObject])
```

72

```
(setGCmessage
  [LAMBDA NIL                                              (* D.Thompson " 8-Apr-80 09:21")

          (* * This routine controls the garbage collection message. It resets the message pieces according to the value of
          the profile entry GarbageCollectionMessage.)
```

```
    (SELECTQ (UserProfile 'GarbageCollectionMessage)
             ((Normal On True Yes)
               (if (InterLispGCGAG (UserProfile 'GarbageCollectionPages))
                   then T
                 else NIL))
             ((False No None Off)
               (if (InterLispGCGAG NIL)
                   then T
                 else NIL))
             (Compact (PROG1 (if (InterLispGCGAG T)
                                 then T
                               else NIL)
                             (GCMESS 1 "<<collecting ")
                             (GCMESS 2 "...")
                             (GCMESS 3)
                             (GCMESS 4)
                             (GCMESS 6 " pages left")
                             (GCMESS 7 ">>
")))
             (PROGN (Unexpected 'GCMessageCategory)
                    (if (InterLispGCGAG T)
                        then T
                      else NIL])
```

73

```
(setGCpages
  [LAMBDA NIL                                          (* D.Thompson " 6-Jul-79 09:10")
    (GCMESS 5 (UserProfile 'GarbageCollectionPages))
    (setGCmessage])
```

74

```
(transcript
  [LAMBDA (fs internalCall)                            (* R.Bates " 6-Aug-80 10:12")
    (PROG (HELPFLAG filename temp outfile)
          (TRANSCRIPTFILE←'GiveMessage)
          (ERSETQ (PROGN filename←(if internalCall
                                      then 'ON
                                    else (U-CASE fs:1))
                         outfile←(if filename='ON or fs=NIL
                                     then (PACKFILENAME 'DIRECTORY (if internalCall
                                                                       then (DIRECTORYNAME NIL T)
                                                                     else (DIRECTORYNAME T T))
                                                        'NAME
                                                        (UserProfile 'TranscriptFileName)
                                                        'EXTENSION
                                                        (DATE 67371008))
                                   elseif filename='OFF
                                     then TRANSCRIPTFILE←NIL
                                          NIL
                                   else (PACK fs))
                         (if (ERSETQ temp←(DRIBBLE outfile))=NIL
                             then (RETURN NIL)
                           else (if temp
                                    then (printout T .TAB0 0 "Transcript file" , temp , "is closed."
                                                   T)
                                         (TRANSCRIPTFILE←NIL)))
                         (if filename~='OFF
                             then (printout T .TAB0 0 "Transcript file" , (DRIBBLEFILE)
                                            T "is open in the AFFIRM system" , AFFIRMMAKESYSFILE T)
                                  TRANSCRIPTFILE←(DRIBBLE (DRIBBLEFILE)
                                                          T T])
```

75

```
(UserEdit
  [LAMBDA (List ReadTable DontAbort)                   (* D.Thompson "23-Jul-80 13:12")
    (PROG (Dfile file temp)
          (file←(OPENFILE 'TEMP.EDITOR 'OUTPUT))
          (PrettyPrintCommand List file ReadTable)
          (do temp←(CallEditor file 'File) repeatuntil temp~='ABORTED or DontAbort=NIL)
          (if temp='ABORTED
              then (CLOSEF? file)
                   (DELFILE file)
                   (AffirmError))
```

```
        (if temp~=file
            then (CLOSEF? file)
                 (DELFILE file)
                 file←temp)
        (PROG [(ERRORTYPELST ('((16 (PROGN (SETQ BREAKCHK NIL)
                                           (SETQ PRINTMSG NIL)
                                           (RETFROM ERRORPOS (QUOTE fixstopfixstop)
                                                    T]
              (temp←(RATOMS 'fixstopfixstop file ReadTable)))
        (CLOSEF? file)
        (DELFILE file)
        (if (DRIBBLEFILE)
            then (printName temp 'FixedCommand))
        (RETURN temp])
)
(DEFINEQ
```

76

```
(print
  [LAMBDA (ratoms)                                    (* D.Thompson "21-Oct-80 18:00")

        (* * This routine parses and decodes the input for the print command, and then calls printHelper with the result.
        -
        An AFFIRM command function.)


    (PROG (key lckey lA type)
          (key←ratoms:1)                              (* may be NIL)
          (lckey←(L-CASE key))
          (pop ratoms)
          (SELECTQ (lckey←(AFFIRMSpellingCorrect lckey KnownNames:PrintObjects NIL T T))
                   (? (printHelper 'known '(PrintObjects)))
                   ((assumptions BadEquations IH implist known names next original prop result status
                                 uses variables)
                     (printHelper lckey ratoms))
                   ((axiom declare defn. exception interface lemma lhs macro needs parts precondition
                           program schema)
                     (printHelper 'parts <lckey !ratoms>))
                   ((both named proof)
                     (SELECTQ ratoms:1
                              (list (pop ratoms)
                                    lA←T)
                              (nolist (pop ratoms)
                                      lA←NIL)
                              (lA←(UserProfile 'ListAppliedExprs T)))
                                                      (* lA is listApplied)
                     (printHelper lckey ratoms lA))
                   (file (TypeFile ratoms))
                   (history (PrettyPrintHistory ratoms))
                   (type (printHelper 'type ratoms:1 ratoms::1))
                   (if type←(CheckForType key NIL 'OK)
                       then (AffirmError <"The preferred syntax is print type" type "." > 'mild)
                            (printHelper 'type type ratoms)
                     elseif (FIXP key) or (DecodeName key 'optional)
                       then (AffirmError "The preferred syntax is print prop or print node." 'mild)
                            (printHelper 'prop <key ! ratoms>)
                     else (AffirmError <"Unknown key for print:" key
                                       "Type 'print ?;' for a list of options."
                                       >])
```

77

```
(printName
  [LAMBDA (objects category)                          (* D.Thompson "15-Jan-81 15:30")

        (* * This routine outputs the Annotating mode forms.)


    [SELECTQ category
             [AnnotationFlag (if Annotating
                                 then (Dprintout T .PARA 0 0 (MKLIST objects]
             (ElseWord (if Annotating
                           then                       (* first, print terminal version.
                                                      Then print transcript version.)
                                (Dprintout T .TAB objects "else" ,)
                                (Dprintout NIL T AtSign "else" LeftSqBracket RightSqBracket)
                      else                            (* just print it out, normal case)
                           (printout NIL .TAB objects "else" ,)))
```

```
            (ExecPrompt (printout NIL .PARA 0 0 (MKLIST objects))
                        (if Annotating
                            then (Dprintout NIL "@UCmd" LeftSqBracket)))
            (FixedCommand (if Annotating
                              then (Dprintout NIL "@UCmd" LeftSqBracket))
                          (Dprintout NIL # (PrettyPrintCommand (MKLIST objects)
                                                               NIL PascalReadTable))
                          (if Annotating
                              then (Dprintout NIL RightSqBracket)))
            (IfWord (if Annotating
                        then                                   (* first, print terminal version.
                                                               Then print transcript version.)
                            (Dprintout T "if" ,)
                            (Dprintout NIL AtSign "if" LeftSqBracket RightSqBracket)
                    else                                       (* just print it out, normal case)
                         (printout NIL "if" ,)))
            (ProofDisplayCommand (if Annotating and objects=AtSign
                                     then (Dprintout NIL AtSign))
                                 (printout NIL .PARA 0 0 (MKLIST objects)))
            (SystemPrintedCommand (if Annotating
                                      then (Dprintout NIL "@SCmd" LeftSqBracket))
                                  (printout NIL .PARA 0 0 (MKLIST objects))
                                  (if Annotating
                                      then (Dprintout NIL RightSqBracket)))
            (ThenWord (if Annotating
                          then                                 (* first, print terminal version.
                                                               Then print transcript version.)
                              (Dprintout T .TAB objects "then" ,)
                              (Dprintout NIL T AtSign "then" LeftSqBracket RightSqBracket)
                      else                                     (* just print it out, normal case)
                           (printout NIL .TAB objects "then" ,)))
            (TypeListing (if Annotating
                             then (Dprintout NIL AtSign objects "[TypeListing]")))
            (TypeName (if Annotating
                          then (Dprintout NIL "@TypeName" LeftSqBracket))
                      (printout NIL .PARA 0 0 (MKLIST objects))
                      (if Annotating
                          then (Dprintout NIL RightSqBracket)))
            (PROGN (Unexpected 'PrintObject)
                   (printout NIL .PARA 0 0 (MKLIST objects]
    NullString])
```

78

```
(PrintNews
  [LAMBDA (Date)                                               (* R.Erickson "29-Jan-81 15:01")

        (* * Prints any news files created since Date, a number. Also notifies user if Knownbugs has been written since
        Date. Sets the global NewsDate. Should be called once per session.)


    (PROG (Files knownFile)
          (if ~(NUMBERP Date)
              then Date←0)
          (NewsDate←(IDATE))

          (* * Look up news files, compute write dates)


          (Files←(DIRECTORY (PACKFILENAME 'DIRECTORY PV\DIRECTORY 'NAME 'NEWS)
                            '(COLLECT)))
          (Files←(for (file date) in Files collect <file date (FILENAMEFIELD file 'EXTENSION) >
                    when (IGEQ date←(GETFILEINFO file 'ICREATIONDATE)
                               Date)))                         (* Files is "(...(name createdate extension)...)")
          (if Files
              then [SORT Files (FUNCTION (LAMBDA (x y)
                                         (x:2 lt y:2]          (* increasing time)
                   (for triple in Files bind file do (printout T .TAB0 0 Stars T)
                                                     (if (NUMBERP triple:3) and triple:3 gt 0
                                                         then (printout T
                                                                        "News especially for version "
                                                                        triple:3 " of AFFIRM:" T)
                                                              (* regardless of current version)
                                                              (* I have to use IGREATERP rather than test ~ = 0 since
                                                              DWIM has a bug)
                                                                )
                                                     (file←triple:1)
                                                     (file←(OPENFILE file 'INPUT))
                                                     (COPYBYTES file T)
                                                     (CLOSEF file))
```

```
            (printout T .TAB0 0 Stars T))
        (knownFile←(PACKFILENAME 'DIRECTORY PV\DIRECTORY 'BODY 'KNOWNBUGS.TXT))
        (if (INFILEP knownFile) and (IGEQ (GETFILEINFO knownFile 'IWRITEDATE)
                                          Date)
            then (printout T .TAB0 0 Stars T
            "We've documented more Affirm bugs since your last session.
Please see message file"
                          , knownFile T])
```

79

(printProofHeader

```
  [LAMBDA NIL                                          (* R.Erickson "16-Oct-79 21:00")
    (printout NIL T "proof tree: " T])
```

80

(printWhat?

```
  [LAMBDA (ratoms defaultitem)                         (* R.Erickson "22-Oct-79 18:18")

          (* * return list of nodes)


    (if ~ratoms
        then ratoms← <defaultitem>)
    (for r in ratoms join (SELECTQ r
                                  (                    (*)
                                    <CurrentNode>)
                                  (T < (GetNode CurrentTheorem:prop#)
                                    >)
                                  (theorems (for t in (SortRecency Theorems)
                                               collect (ThmToNode t)))
                                  ((unproved unproven)
                                    (for t in (SortRecency Theorems) unless (Proved? t)
                                       collect (ThmToNode t)))
                                  (<(GetNode r)
                                    >])
)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* AFFIRM Executive Tables and Data) ]

[DECLARE: EVAL@COMPILE

(RECORD AFFIRMEnvironment (BreakAccess LineLength GCMessage GCPages HistoryLength))

(RECORD LineDeleteMessages (CommandEnv NormalEnv ParameterEnv))

(RECORD TerminalEnvironment (LineBuffering EchoMode EchoControls DeleteControls DeleteEchoMode))
]

(RPAQ Annotating NIL)

(RPAQ AnnotationOpen NIL)

(RPAQ BatchAnswers NIL)

(RPAQ BatchMode NIL)

(RPAQ BootStrapping NIL)

(RPAQ CommandTerminator ';)

(RPAQ CurrentTerminalEnvironment NIL)

(RPAQ DeleteControlTypes ' (LINEDELETE 1STCHDEL NTHCHDEL POSTCHDEL EMPTYCHDEL))

(RPAQ FreezeFileName 'Frozen-AFFIRM)

(RPAQ GripeSubjectMaxLength 36)

(RPAQQ HistorylessCommands (stop abort exec quit))

(RPAQ ImplicitE NIL)

(RPAQ LineDeleteMessage (create LineDeleteMessages CommandEnv← (DELETECONTROL 'LINEDELETE)
                               NormalEnv←
                               (DELETECONTROL 'LINEDELETE)
                               ParameterEnv← "<
~>"))

(RPAQ LispCommands ' (e ImplicitE))

(RPAQ LispFormStarters <LeftSqBracket LeftParenthesis>)

(RPAQ MessageNoTranscriptGiven NIL)

(RPAQ OldBreakAccess NIL)

(RPAQ OriginalCommandWord NIL)

(RPAQ OwnReadCommands <CommandTerminator>)

(RPAQ PascalReadTable (COPYREADTABLE 'ORIG))

(RPAQ SpecialReadCommands ' (@ e fix ImplicitE note profile redo stop undo))

(RPAQ Tab 7)

(RPAQ Terminal T)

(RPAQ UnBufferedCommands ' ((invoke let put replace suppose)
                            (apply augment axiom discard eval lemma schema try use)))

(RPAQ UsingTed NIL)

(RPAQ CurrentBreakCharacters (APPEND (GETBRK PascalReadTable)
                                     (GETSEPR PascalReadTable)))

(RPAQ NormalPrompt "U:")

(RPAQ BatchErrorPrompt (CONCAT "##" Blank NormalPrompt))

(RPAQ Prompt NormalPrompt)
(SETBRK '
        (4 5 6 14 16 17 18 19 20 21 28 29 34 30 33 38 40 41 42 43 44 45 46 47 58 59 60 61 62 64 91 93
           94 123 124 125 126)
        NIL PascalReadTable)

(PUTPROPS denote AFFIRMCommand denote)
```

```
(PUTPROPS normint AFFIRMCommand normint)

(PUTPROPS swap AFFIRMCommand swap)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* Redefined INTERLISP functions) ]

(DEFINEQ

                                                                    81

(AffirmGCGAG
  [LAMBDA (param)                                   (* D.Thompson " 4-Oct-79 08:19")
    (if param
        then (if param~=T
                 then (printout T T "GCGAG called with param =" , param "." T))
             (setGCmessage)
      elseif (InterLispGCGAG NIL)
        then T
      else NIL])

                                                                    82

(AffirmPRINTBELLS
  [LAMBDA NIL                                       (* D.Thompson " 6-Feb-80 09:04")

          (* * This routine encapsulates the interlisp PRINTBELLS routine to avoid control-G's getting into the transcript;
          they only go to the terminal.)


    (Dprintout T # (InterLispPRINTBELLS])
)
(if (FGETD 'InterLispGCGAG)
    else
    (MOVD 'GCGAG 'InterLispGCGAG))
(if (FGETD 'InterLispSort)
    else
    (MOVD 'SORT 'InterLispSort))
(if (FGETD 'InterLispPRINTBELLS)
    else
    (MOVD 'PRINTBELLS 'InterLispPRINTBELLS))
(MOVD 'AffirmGCGAG 'GCGAG)
(MOVD 'AffirmSORT 'SORT)
(MOVD 'AffirmPRINTBELLS 'PRINTBELLS)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* File / File Name Handling)]

```
(DEFINEQ
```

83

```
(getFileName
  [LAMBDA (firstTry defaultFileName askUser IO dontTellUser)
                                                    (* D.Thompson " 5-Sep-80 20:06")
    (PROG (fileName)
          (RETURN (if firstTry=Terminal
                      then Terminal
                    else (SELECTQ IO
                                  ((INPUT NIL)
                                    (if firstTry~=NIL and (INFILEP (makeFileName firstTry))
                                      elseif ~askUser
                                        then defaultFileName~=NIL
                                             and (INFILEP (makeFileName defaultFileName))
                                      else (if firstTry=NIL or dontTellUser
                                               else (printout NIL .TAB0 0 "Can't find file" .
                                                              (makeFileName firstTry)
                                                              T))
                                           (if fileName←(TTYINFILE (if askUser=T
                                                                       then NIL
                                                                     else askUser)
                                                                   T
                                                                   (makeFileName defaultFileName))
                                               ~='ABORTED
                                               then fileName)))
                                  ((OUTPUT T)
                                    (if firstTry~=NIL and (OUTFILEP (makeFileName firstTry))
                                      elseif ~askUser
                                        then defaultFileName~=NIL
                                             and (OUTFILEP (makeFileName defaultFileName))
                                      else (if firstTry=NIL or dontTellUser
                                               else (printout NIL .TAB0 0 "Can't use file" .
                                                              (makeFileName firstTry)
                                                              T))
                                           (if fileName←(TTYOUTFILE (if askUser=T
                                                                        then NIL
                                                                      else askUser)
                                                                    T
                                                                    (makeFileName defaultFileName))
                                               ~='ABORTED
                                               then fileName)))
                                  (Unexpected 'IOIndicator])
```

84

```
(makeFileName
  [LAMBDA (fileName)                                (* D.Thompson " 5-Sep-80 15:08")

          (* * returns either NIL or an atom representing the file name.)


    (PROG (packedName)
          (RETURN (U-CASE (if (LISTP fileName)
                              then (if (AFFIRMSpellingCorrect (firstElement fileName)
                                                              <'BODY ! FILENAMEFIELDS> NIL T T)
                                       and (NLSETQ packedName←(PACKFILENAME fileName))
                                       then packedName
                                     else (PACK fileName))
                            elseif (LITATOM fileName)
                              then fileName])
```

85

```
(markFileNameAsUsed
  [LAMBDA (fileName category)                       (* D.Thompson "17-May-79 15:35")
    (PROG NIL
      NIL])
```

86

```
(OKtoUseCOMS?
  [LAMBDA (fileName)                                (* D.Thompson "17-May-79 15:17")
    (PROG NIL
```

```
        (RETURN T])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (4123 68558 (AddCommand 4135 . 5890) (AddCommandToHistory 5894 . 6808) (
AddParametersToHistory 6812 . 8275) (AffirmExec 8279 . 8709) (Annotating 8713 . 8906) (AppendChar 8910
 . 9520) (CheckForCurrentType 9524 . 9732) (CheckForType 9736 . 11668) (ClearTerminalBuffer 11672 .
12187) (CloseAnnotation 12191 . 12567) (DefinedAFFIRMCommand 12571 . 12908) (DeleteCommand 12912 .
14717) (DetermineObjectClass 14721 . 16221) (DTVS 16225 . 17866) (Dtvs 17870 . 19569) (EatComments
19573 . 20089) (EndMonitorCycle 20093 . 20735) (ErrorInFile 20739 . 21435) (GetArgumentsFromList 21439
 . 22293) (HistoryWarning 22297 . 22569) (TopLevel? 22573 . 22704) (getAssocCommandFcn 22708 . 23305)
(GetCommand 23309 . 24888) (getElements 24892 . 25944) (GetParameters 25948 . 26601) (IgnoreExcess
26605 . 26927) (InitializeCommandLoop 26931 . 27339) (LoadUserInitializationFile 27343 . 27794) (
MissingECommand? 27798 . 28386) (nodeExpression 28390 . 29345) (ObjectClassOfParameters 29349 . 30613)
 (OpenAnnotation 30617 . 30832) (PerformAutoAppliedCommands 30836 . 33121) (PleaseDeclare 33125 .
34115) (Plural 34119 . 36298) (PopTypeStack 36302 . 36806) (PrintEnd 36810 . 37192) (PrintPrompt 37196
 . 38672) (ProcessCommand 38676 . 56238) (ProcessCommandAbort 56242 . 57377) (ReadCommandFile 57381 .
57563) (readParams 57567 . 58028) (ReadRestOfLine 58032 . 58773) (ReadSpecialCommandParameters 58777 .
 60262) (readString 60266 . 60950) (RemoveChar 60954 . 61367) (RestoreTerminalEnvironment 61371 .
62071) (SaveTerminalEnvironment 62075 . 62554) (SetEventCompletionFlag 62558 . 62896) (Singularize
62900 . 63103) (SkipOverComments 63107 . 63374) (SkipToRealChars 63378 . 63588) (StartMonitorCycle
63592 . 63728) (SuppressAdvisedPrint? 63732 . 65848) (UpdateLineDeleteMessage 65852 . 66579) (
ValidateFileName 66583 . 67841) (VerbInString 67845 . 68555)) (68626 86990 (AffirmCompile 68638 .
69953) (AffirmLoad 69957 . 70524) (AffirmNews 70528 . 71419) (AffirmSave 71423 . 72750) (batch 72754 .
 73637) (EstablishInterLispEnvironment 73641 . 74619) (gripe 74623 . 78373) (limboCommand 78377 .
78754) (LowerInterLisp 78758 . 79098) (monitor 79102 . 79374) (redo 79378 . 81337) (renumber 81341 .
82250) (RestoreAFFIRMEnvironment 82254 . 82808) (save 82812 . 83684) (setGCmessage 83688 . 84661) (
setGCpages 84665 . 84857) (transcript 84861 . 86050) (UserEdit 86054 . 86987)) (86992 94571 (print
87004 . 88862) (printName 88866 . 91653) (PrintNews 91657 . 93740) (printProofHeader 93744 . 93907) (
printWhat? 93911 . 94568)) (96834 97525 (AffirmGCGAG 96846 . 97182) (AffirmPRINTBELLS 97186 . 97522))
(97931 100481 (getFileName 97943 . 99571) (makeFileName 99575 . 100168) (markFileNameAsUsed 100172 .
100321) (OKtoUseCOMS? 100325 . 100478)))))
STOP
```