<AFFIRM>UTILITY..78 30-Sep-81 15:44:36

```
(FILECREATED "26-Sep-81 20:07:16" <AFFIRM>UTILITY..78 117991

     changes to:  UTILITYCOMS

     previous date: "26-Sep-81 18:31:49" <AFFIRM>UTILITY..77)


(PRETTYCOMPRINT UTILITYCOMS)

(RPAQQ UTILITYCOMS [(* 1 Records (except XIVUS))
        (RECORDS * CEVALRECORDS)
        (RECORDS IfExpression)
        (RECORDS CommandCategories)
        (RECORDS * DTVSRECORDS)
        (RECORDS * LOGICRECORDS)
        (RECORDS * NEWEQUALRECORDS)
        (RECORDS * PATTERNCOMPILERRECORDS)
        (RECORDS PropStorage)
        (RECORDS * REPLACERECORDS)
        (RECORDS * SKOLEMRECORDS)
        (RECORDS * TREERECORDS)
        (RECORDS UnexpectedList UserProfileEntryInformation)
        (RECORDS RuleTypes LhsEntry)
        (RECORDS PROPS)
        (RECORDS KnownNames)
        (* 2 Printing by normal AFFIRM functions)
        (VARS (SuppressCompilerMsgs)
              (ListingFlag NIL)
              ListingKinds
              (ReportWord "using ")
              (RuleLHSPercentage 65)
              (TRANSCRIPTFILE (QUOTE GiveMessage))
              TEDFILE)
        (* 3 PrettyPrinter and Parser vars)
        (VARS (#Current\LINELENGTH (LINELENGTH))
              (DOTS\FLAG NIL)
              (ComingBackFromSYSOUT NIL)
              (FewPagesMessage NIL))
        (PROP GLOBALVAR CANON DOTS\FLAG FEWER\PARENS LESS\THAN\FLAG PARSER\PROMPT)
        (PROP OriginalName =\Interface ExpressionWithType)
        (* 4a AFFIRM command-level vars)
        (VARS (AutoCases NIL)
              (CurrentEventNumber NIL)
              (Completion T)
              (CANON NIL)
              (CautiousCompletion NIL)
              (DTVSCANON NIL)
              (DtvsTimeFlag T)
              (LessOutputDesired NIL)
              (REPORTFLAG NIL)
              (Timer NIL))
        (* 5 Executive, TheoremProver, RewriteRule internal state)
        (VARS (Assumed NIL)
              (BadEquations NIL)
              (CurrentContext NIL)
              (CurrentType NIL)
              (Denied NIL)
              (FileTypes (QUOTE (Compiled Saved Source)))
              (InvokedDefinitions NIL)
              (PrettyNorm NIL)
              (ProvingMode NIL)
              (PV\Library 'PVLibrary)
              (RuleHistory NIL)
              (RuleTypes (create RuleTypes))
              (RuleList NIL)
              (TypeStack NIL)
              (Unchecked NIL)
              (UseRules T)
              (UsingRuleList NIL))
        (* 6a Mnemonic constants)
        (VARS PROPOP IH1OP IH2OP (IEXT (QUOTE Interface))
              (EQVOP 'EQV\Boolean)
              (ErrorOp 'ERROR)
              (IFOP 'IfThenElse)
              (QOP 'Qexpression)
              (QUOTEOP 'QUOTE)
              (TRUE 'TRUE)
              (FALSE 'FALSE))
        (* 6b Character Literal Constants)
        (VARS (Ampersand (QUOTE &))
              (Asterisk '*)
```

```
        (AtSign '@)
        (Blank (FCHARACTER 32))
        (CR (FCHARACTER 31))
        (Colon ':)
        (Comma ',)
        (DoubleQuote '%")
        (EqualSign '=)
        (Escape (FCHARACTER 27))
        (ExclamationPoint '!)
        (Hyphen '-)
        (LeftCurlyBracket '{)
        (LeftParenthesis '%()
        (LeftSqBracket '%[)
        (LF (CHARACTER 10))
        (NullString "")
        (Period '.)
        (PercentSign '%%)
        (RightCurlyBracket '})
        (SingleQuote '')
        (Slash '/)
        Stars
        (TabChar (FCHARACTER 9))
        (TypeSeparator '\)
        (QuestionMark '?)
        (Ellipses '...)
        (RightParenthesis '%))
        (RightSqBracket '%])
        (SemiColon ';))
(* 7 Interfaces, IsConstant)
(VARS FALSE\Interface TRUE\Interface)
(PROP IsConstant FALSE TRUE)
(* 8a various utility functions)
(FNS * AtomFNS)
[VARS ((UCTLArray (CONS (HARRAY 50)
                        50]
(FNS * ExtensionFNS)
(FNS * ForkFNS)
(FNS * HereForCompileFNS)
(FNS * InitFNS)
(FNS * ListFNS)
(FNS * PrintFNS)
(FNS * XevalFNS)
(* 8b leftover utility functions - should be moved elsewhere)
(FNS * UTILITYFNS)
(VARS (AFFIRMVERSION 0)
      SearchBadFile TEMPRELEASECOMMENTS VERSIONFILE (AFFIRMINITFIRSTTIME NIL)
      [DebugMinfs (QUOTE ((1 . 10000)
                          (4 . 512)
                          (5 . 512)
                          (8 . 10000)
                          (9 . 512)
                          (12 . 1000)
                          (16 . 512)
                          (18 . 3000)
                          (24 . 512)
                          (28 . 512)
                          (30 . 512]
      (ExtensionHashArray < (HARRAY 200)
                          ! 1.5>)
      MinfsLevels
      (ScratchList (for I from 1 to 35 collect NIL))
      (ShortenHashArray SYSHASHARRAY)
      SYSTEMSTATECOMS)
(P (/MOVD (QUOTE Affirm/ADDPROP)
          (QUOTE /ADDPROP)))
(* LIMBO: apparently useless)
(VARS (any 'any)
      (IMPLIST NIL)
      (Prompt "U: "))
(* 9 Error handling, interrupts)
(FNS * AFFIRMErrorFNS)
(VARS (AskUserToDefine T)
      (BADFILENAMES NIL)
      (DTVSCOMPLAIN T)
      (DWIMUSERFN T)
      HELPDEPTH
      (NOSAVESETVARS (QUOTE (ZapKind KnownTypes)))
      (UnexpectedCategories (QUOTE (AFFIRMObject File HelpTopic LineDelete Need PrintObject
                                                 ProfileEntryName UnexpectedCategory)))
      (Unexpected (for category in UnexpectedCategories collect
                     (create UnexpectedList Category ← (U-CASE category)
```

```
                                    Elements ← NIL)))
      (UserInterrupt NIL))
(ALISTS (ERRORTYPELST 23 42))
[ADDVARS (BREAKRESETFORMS (AffirmBreak (QUOTE ENTERING]
(P (INTERRUPTCHAR 24 'UserInterrupt))
(* 10a System gens, sysouts, files, user initials)
(FNS * SYSTEM\MAKERFNS)
[VARS (AFFIRMSYSOUTFILE NIL)
      (CannedMessagesSeen NIL)
      (COMPILEHEADER " compiled for AFFIRM on ")
      (MKSWAPSIZE 0)
      (PRETTYHEADER " file created for AFFIRM on ")
      [PV\BASIS\FILES (QUOTE ((AFFIRM . Boolean)
                              (AFFIRM . BUILTIN)
                              (AFFIRM . Integer)
                              (AFFIRM . TypeParameter)
                              (AFFIRM . Induction)
                              (AFFIRM . ProcedureCall]
      (PV\DIRECTORY 'AFFIRM)
      [PV\SYSTEM\FILES (QUOTE ((AFFIRM . AFFIRMEXEC)
                               (AFFIRM . CEVAL)
                               (AFFIRM . FORMULAIO)
                               (AFFIRM . HELP)
                               (AFFIRM . INFIXPRINT)
                               (AFFIRM . LOGIC)
                               (AFFIRM . PARSER)
                               (AFFIRM . PARSERHELPER)
                               (AFFIRM . PASCAL)
                               (AFFIRM . PROFILE)
                               (AFFIRM . REQUEST)
                               (AFFIRM . REWRITERULE)
                               (AFFIRM . SPECIFICATION)
                               (AFFIRM . SUFFICIENT)
                               (AFFIRM . TEDIT)
                               (AFFIRM . THEOREMPROVER)
                               (AFFIRM . TREE)
                               (AFFIRM . VCGEN)
                               (AFFIRM . XEVAL]
      (PVMAINTAINERS (QUOTE (DTHOMPSON]
(ADDVARS (INITIALSLST (BAKER . D.Baker)
                      (DTHOMPSON . D.Thompson)
                      (ERICKSON . R.Erickson)
                      (GERHART . S.Gerhart)
                      (LONDON . R.London)
                      (MUSSER . D.Musser)
                      (RBATES . R.Bates)
                      (TAYLOR . D.Taylor)
                      (WILE . D.Wile)))
[P (if ~ (EDITFINDP AFTERSYSOUTFORMS 'AffirmSysoutInit)
       then
       (/SET (QUOTE AFTERSYSOUTFORMS)
             (APPEND AFTERSYSOUTFORMS (LIST (QUOTE (INIT\BACKSPACE))
                                            (QUOTE (AffirmSysoutInit !VALUE]
(* 10b Name definitions -- predefined, hidden, reserved, known. Depends on variables in 10a)
(VARS PredefinedNames
      [HiddenNames (create KnownNames AFFIRMObjects ← (QUOTE (Unexpected))
                           Commands ←
                           (QUOTE (; batch copy enter ImplicitE keep lemma macro monitor
                                     program rule show types))
                           FileTypes ← (QUOTE (OldCompiled OldSaved))
                           Types ← (QUOTE (BUILTIN Induction TypeParameter]
      [ReservedNames (create KnownNames Types ← (QUOTE (Interface Schema]
      (KnownNames (create KnownNames PrintObjects ←
                          (APPEND (fetch PrintObjects of PredefinedNames)
                                  (COPY (fetch TypeParts of PredefinedNames)))
                          USING PredefinedNames)))
(* 11 Advice)
(PROP ARGNAMES PRINT)
(ADVISE (PRINT IN COMPILE1A)
        (PRINT IN LAPBLOCK))
(ADVISE LOGOUT SAVESET SUBSYS)
(* 12 XIVUS)
(VARS NEGINF POSINF)
(VARS * NEWNAMESVARS)
(RECORDS ArrayAccess ArrayWrite Assign HeapOrFileAccess HeapOrFileWrite RecordAccess
         RecordWrite priorityProp priorityRecord)
(RECORDS * UTILITYRECORDS)
(P (#INIT\XEVAL))
(* 13 compiled files and CLISP)
(P (LOADCOMP (QUOTE <affirm>parser))
   (CLISPDEC (QUOTE FAST)))
```

```
        (VARS (UPDATEMAPFLG))
        (* 14 LISPX)
        (ADDVARS (LISPXMACROS (exec (PROGN (OR (NLSETQ (SETQ LASTEXEC (SUBSYS LASTEXEC)))
                                               (SETQ LASTEXEC (SUBSYS)))
                                           LASTEXEC))
                              (qu (LOGOUT))
                              (quit (LOGOUT))
                              (TCOPY (for x in LISPXLINE collect (TypeFile x)))
                              (XED (SetupXed)))
                 (LISPXCOMS exec qu quit TCOPY XED))
        (LISPXMACROS EXEC TECO ok)
        (* 15 Macros, info)
        (MACROS CONSIT Dprintout EQCAR EvalFormWithOutDribble IfThenElse NoSpreadApply* PRINTHELP
                PRINTLINES TranslateApply UCI\SUBST UserProfile)
        (TEMPLATES ADDTOLIST DROPFROMLIST Dprintout MapExpr Minimize NoSpreadApply* TranslateApply
                   TranslateApply)
        (PROP INFO IfThenElse PRINTLINES EvalFormWithOutDribble MapExpr)
        (GLOBALVARS ADDIO ALLIO ANDIO Ampersand Asterisk AtSign Blank CR Colon Comma DIFFIO DIVIO
                    DoubleQuote EQIO EQVIO Ellipses EqualSign Escape ExclamationPoint FALSE GEIO
                    GLOBALVAR GTIO Hyphen Hyphens IEXT IMPIO INVIO KnownNames LEIO LF LTIO
                    LeftCurlyBracket LeftParenthesis MAXIO MINIO MODIO MULTIO NEGIO NEIO NOTIO
                    NullString ORIO PARSERTRACE PercentSign Period Period QOP QUOTIENTIO QuestionMark
                    RESETVARSLST RightCurlyBracket RightParenthesis SOMEIO SemiColon SingleQuote
                    Slash SquareBracket Stars TRUE TabChar Terminal TypeSeparator
                    (RuleTypes RuleList))
        (DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
                  (ADDVARS (NLAMA EvalFormWithOutDribble PRINTLINES)
                           (NLAML XSet BEEHIVE Report)
                           (LAMA])
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 1 Records (except XIVUS)) ]

(RPAQQ CEVALRECORDS (Equation Form IfRecord))
[DECLARE: EVAL@COMPILE

(RECORD Equation (Operator LHS RHS)
                 Operator ← EQOP)

(RECORD Form (Operator Arg1 Arg2 Arg3 Arg4 Arg5))

(RECORD IfRecord (Operator Test ThenPart ElsePart)
                 Operator ← IFOP IfRecord @(EQ (fetch Operator of DATUM)
                                               IFOP))
]
[DECLARE: EVAL@COMPILE

(RECORD IfExpression (Operator Test ThenPart ElsePart)
                     Operator ← IFOP IfExpression @(AND (LISTP DATUM)
                                                        (EQ (fetch Operator of DATUM)
                                                            IFOP)))
]
[DECLARE: EVAL@COMPILE

(RECORD CommandCategories (ExecutiveMachine SpecificationMachine TheoremProverMachine
                                            MiscellaneousMachine ProofMaintenance))
]

(RPAQQ DTVSRECORDS (Proof TheoremList lemmarecord))
[DECLARE: EVAL@COMPILE

(RECORD Proof (Status . Assumptions))

(RECORD TheoremList (NumberPreds PredicateList ProofList))

(ATOMRECORD lemmarecord (lemma))
]

(RPAQQ LOGICRECORDS ((ATOMRECORD (NewSymbolMapping))))
[DECLARE: EVAL@COMPILE
(ATOMRECORD (NewSymbolMapping))
]

(RPAQQ NEWEQUALRECORDS ((ATOMRECORD (EQOP))
                        (ATOMRECORD (EqualOp))))
[DECLARE: EVAL@COMPILE
(ATOMRECORD (EQOP))
(ATOMRECORD (EqualOp))
]

(RPAQQ PATTERNCOMPILERRECORDS (FunDef SubPair Expression ExpressionWithType PROPS RewriteRule))
[DECLARE: EVAL@COMPILE

(RECORD FunDef (NIL Args . Body))

(RECORD SubPair (OLD . NEW))

(RECORD Expression (Operator . Arguments))

(TYPERECORD ExpressionWithType (Expression Type))

(ATOMRECORD PROPS (Axioms DeclaredType Distinct Infix InterfaceRules IsConstant LocalDeclarations
                          Needs NoChange OriginalName PrimaryLHSides Programs RepRules Rules))

(RECORD RewriteRule (RewriteOp LHS RHS)
                    RewriteOp ←(QUOTE Equal))
]
[DECLARE: EVAL@COMPILE

(RECORD PropStorage (#ToExpr ExprTo# Top#)
                    Top# ← 0 ExprTo# ←(LIST (CONS TRUE 0))
                    #ToExpr ←(LIST (CONS 0 TRUE)))
]

(RPAQQ REPLACERECORDS ((ATOMRECORD (EQOP))))
[DECLARE: EVAL@COMPILE
(ATOMRECORD (EQOP))
]

(RPAQQ SKOLEMRECORDS (Qexpression))
[DECLARE: EVAL@COMPILE
```

```
(TYPERECORD Qexpression (given find (freevars . free)
                            expr)
                        freevars ←(QUOTE freevars))
]

(RPAQQ TREERECORDS (Node Theorem Transformation Propositions annotation Circularity groupmember))
[DECLARE: EVAL@COMPILE

(RECORD Node (prop# parents trans annotation . oldtrans)
            [TYPE? (AND (LISTP DATUM)
                        (FIXP (fetch prop# of DATUM))
                        (OR (NOT (fetch parents of DATUM))
                            (LISTP (fetch parents of DATUM])

(RECORD Theorem (prop# status factsused usedby lastchain recency . leaves)
                [TYPE? (AND (FIXP (fetch prop# of DATUM))
                            (LITATOM (FETCH status OF DATUM])

(RECORD Transformation (title children labels annotation . uses)
                       (RECORD title (command parameters event#))
                       [TYPE? (AND (LISTP DATUM)
                                   (LISTP (fetch title of DATUM]
                       (CREATE (CompressTransform DATUM)))

(ACCESSFNS Propositions [name (CDR (FASSOC DATUM Prop#sToNames))
                              (COND
                                (Prop#sToNames (/PUTASSOC DATUM NEWVALUE Prop#sToNames))
                                (T (/SET (QUOTE Prop#sToNames)
                                         (LIST (CONS DATUM NEWVALUE])

(RECORD annotation (usernote . systemnote))

(RECORD Circularity (circulartype . circularity))

(TYPERECORD groupmember (grpmprop grpmname grpmannotation grpmstat))
]
[DECLARE: EVAL@COMPILE

(RECORD UnexpectedList (Category Elements))

(RECORD UserProfileEntryInformation (ProfileEntryName DisplayString PossibleValues DefaultValue
                                                       SubKeys Conditions AssociatedVariables))
]
[DECLARE: EVAL@COMPILE

(RECORD RuleTypes (Axiom Lemma Define Schema . OtherRules)
                  Axiom ←(QUOTE axiom)
                  Lemma ←(QUOTE lemma)
                  Define ←(QUOTE defn.)
                  Schema ←(QUOTE schema)
                  (RECORD OtherRules (Automatic Program Precondition Exception Interface)
                          Automatic ←(QUOTE automatic)
                          Program ←(QUOTE program)
                          Precondition ←(QUOTE precondition)
                          Exception ←(QUOTE exception)
                          Interface ←(QUOTE interface)))

(RECORD LhsEntry (eid . elhs))
]
[DECLARE: EVAL@COMPILE

(ATOMRECORD PROPS (Axioms DeclaredType Distinct Infix InterfaceRules IsConstant LocalDeclarations
                          Needs NoChange OriginalName PrimaryLHSides Programs RepRules Rules))
]
[DECLARE: EVAL@COMPILE

(RECORD KnownNames (AFFIRMObjects Arcs Axioms Commands Definitions Directories Files FileTypes Groups
                                  HelpTopics Interfaces Lemmas Nodes PrintObjects ProfileEntries
                                  Schemas Types TypeParts Variables Unexpected Miscellaneous
                                  DiscardOptions))
]
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 2 Printing by normal AFFIRM functions)]

```
(RPAQ SuppressCompilerMsgs NIL)

(RPAQ ListingFlag NIL)

(RPAQQ ListingKinds (decl interface macro defn. axiom schema program precondition exception lemma))

(RPAQ ReportWord "using ")

(RPAQ RuleLHSPercentage 65)

(RPAQQ TRANSCRIPTFILE GiveMessage)

(RPAQQ TEDFILE AFFIRM.COMMAND)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 3 PrettyPrinter and Parser vars) ]

(RPAQ #Current\LINELENGTH (LINELENGTH))

(RPAQ DOTS\FLAG NIL)

(RPAQ ComingBackFromSYSOUT NIL)

(RPAQ FewPagesMessage NIL)

(PUTPROPS CANON GLOBALVAR T)

(PUTPROPS DOTS\FLAG GLOBALVAR T)

(PUTPROPS FEWER\PARENS GLOBALVAR T)

(PUTPROPS LESS\THAN\FLAG GLOBALVAR T)

(PUTPROPS PARSER\PROMPT GLOBALVAR T)

(PUTPROPS =\Interface OriginalName EQ)

(PUTPROPS ExpressionWithType OriginalName :)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 4a AFFIRM command-level vars) ]

(RPAQ AutoCases NIL)

(RPAQ CurrentEventNumber NIL)

(RPAQ Completion T)

(RPAQ CANON NIL)

(RPAQ CautiousCompletion NIL)

(RPAQ DTVSCANON NIL)

(RPAQ DtvsTimeFlag T)

(RPAQ LessOutputDesired NIL)

(RPAQ REPORTFLAG NIL)

(RPAQ Timer NIL)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 5 Executive, TheoremProver, RewriteRule internal state)]

```
(RPAQ Assumed NIL)

(RPAQ BadEquations NIL)

(RPAQ CurrentContext NIL)

(RPAQ CurrentType NIL)

(RPAQ Denied NIL)

(RPAQQ FileTypes (Compiled Saved Source))

(RPAQ InvokedDefinitions NIL)

(RPAQ PrettyNorm NIL)

(RPAQ ProvingMode NIL)

(RPAQ PV\Library 'PVLibrary)

(RPAQ RuleHistory NIL)

(RPAQ RuleTypes (create RuleTypes))

(RPAQ RuleList NIL)

(RPAQ TypeStack NIL)

(RPAQ Unchecked NIL)

(RPAQ UseRules T)

(RPAQ UsingRuleList NIL)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 6a Mnemonic constants)]

```
(RPAQQ PROPOP Prop\Induction)

(RPAQQ IH1OP IH\Induction)

(RPAQQ IH2OP IH\Boolean)

(RPAQQ IEXT Interface)

(RPAQ EQVOP 'EQV\Boolean)

(RPAQ ErrorOp 'ERROR)

(RPAQ IFOP 'IfThenElse)

(RPAQ QOP 'Qexpression)

(RPAQ QUOTEOP 'QUOTE)

(RPAQ TRUE 'TRUE)

(RPAQ FALSE 'FALSE)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 6b Character Literal Constants)]

```
(RPAQQ Ampersand &)

(RPAQ Asterisk '*)

(RPAQ AtSign '@)

(RPAQ Blank (FCHARACTER 32))

(RPAQ CR (FCHARACTER 31))

(RPAQ Colon ':)

(RPAQ Comma ',)

(RPAQ DoubleQuote '%")

(RPAQ EqualSign '=)

(RPAQ Escape (FCHARACTER 27))

(RPAQ ExclamationPoint '!)

(RPAQ Hyphen '-)

(RPAQ LeftCurlyBracket '{)

(RPAQ LeftParenthesis '%()

(RPAQ LeftSqBracket '%[)

(RPAQ LF (CHARACTER 10))

(RPAQ NullString "")

(RPAQ Period '.)

(RPAQ PercentSign '%%)

(RPAQ RightCurlyBracket '})

(RPAQ SingleQuote '')

(RPAQ Slash '/)

(RPAQQ Stars "********")

(RPAQ TabChar (FCHARACTER 9))

(RPAQ TypeSeparator '\)

(RPAQ QuestionMark '?)

(RPAQ Ellipses '...)

(RPAQ RightParenthesis '%))

(RPAQ RightSqBracket '%])

(RPAQ SemiColon ';)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 7 Interfaces, IsConstant) ]

(RPAQQ FALSE\Interface (ExpressionWithType FALSE Boolean))

(RPAQQ TRUE\Interface (ExpressionWithType TRUE Boolean))

(PUTPROPS FALSE IsConstant T)

(PUTPROPS TRUE IsConstant T)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 8a various utility functions)]

```
(RPAQQ AtomFNS (EndinColon EndsColon Skim UCaseToList))
(DEFINEQ
```

1

```
(EndinColon
  [LAMBDA (atm)                                          (* D.Thompson "19-Aug-80 11:06")
    (if (LITATOM atm) and atm and Colon~=(NTHCHAR atm -1)
        then (PACK* atm Colon)
      else atm])
```

2

```
(EndsColon
  [LAMBDA (atom)                                         (* R.Erickson " 1-Oct-79 17:07")
    (GLC atom)=':])
```

3

```
(Skim
  [LAMBDA (symbol char)                                  (* D.Thompson "26-Sep-79 15:38")
    (PACK (DREMOVE (if char=NIL or char=T
                       then Blank
                     else char)
                   (UNPACK symbol])
```

4

```
(UCaseToList
  [LAMBDA (atom list)                                    (* R.Erickson "18-Feb-81 17:05")

        (* * if (U-CASE atom) is in list, returns it, else NIL. We assume list is constant between calls, and all
        upper-case! Also assume atom~ = T! Uses a hash array for speed. Note that U-CASEs which miss are collectable.)

    (PROG (entry)                                        (* UCTLArray is hash array with upper if in list, T if
                                                         not.)
          (RETURN (if entry←(GETHASH atom UCTLArray)
                      then (if entry=T
                               then                      (* known, not on list)
                                    NIL
                             else entry)
                    else entry←(if (U-CASEP atom)
                                   then atom
                                 else (U-CASE atom))
                         (if entry MEMB list
                             then (PUTHASH atom entry UCTLArray)
                           else (PUTHASH atom T UCTLArray)
                                NIL])
)

(RPAQ UCTLArray (CONS (HARRAY 50) 50) NIL)

(RPAQQ ExtensionFNS (ExtendName ExtendUsingList Extension MakeExtension Shorten ShortenLessPrimes))
(DEFINEQ
```

5

```
(ExtendName
  [LAMBDA (Name Extension)                               (* D.Thompson "12-Aug-80 08:38")
    Name←(if Extension='BUILTIN
             then Name
           else (PACK* Name TypeSeparator Extension])
```

6

```
(ExtendUsingList
  [LAMBDA (shorts longs msg)                             (* R.Erickson "22-Sep-81 17:11")
                                                         (* longs a list of atoms. Shorten of which is unique.
                                                         shorts is a list of atoms. msg used for failure)

        (* * For each of shorts. return that long such that (EQ (Shorten long) short) -
```

```
        If such a one is not found, return NIL or, if msg is given, cause an error with that message.)

    (if shorts and (NLISTP shorts)
        then (ExtendUsingList <shorts> longs msg):1
      else (PROG (alist)
                 (alist←(for long in longs collect <(Shorten long) ! long>))
                 (RETURN (for short in shorts bind long when (if long←(ASSOC short alist)::1
                                                             elseif msg
                                                               then (AffirmError < ! msg Colon
                                                                                   (Shorten short)
                                                                                   >)
                                                                    NIL)
                                                    collect long])

                                                                                                    7

(Extension
  [LAMBDA (Name lastonly)                                    (* D.Thompson "12-Aug-80 08:40")
    (if lastonly
        then

          (* * return the LAST extension in the atom Name)

             (PROG (ex ex2)
                   (ex←(Extension Name))
                   (while ex2←(Extension ex) do ex←ex2)
                   (RETURN ex))
      else

          (* * return ALL the extensions in the atom Name)

           (if (GETHASH Name ExtensionHashArray)
             else (for (x stemp) on (DUNPACK Name ScratchList) do stemp← < !! stemp x:1>
                     until x:1=TypeSeparator finally (RETURN (if x::1
                                                                then (PUTHASH Name (PACK stemp)
                                                                              ShortenHashArray)
                                                                     (PUTHASH Name (PACK x::1)
                                                                              ExtensionHashArray)
                                                           else NIL])

                                                                                                    8

(MakeExtension
  [LAMBDA (x ext)                                            (* D.Thompson "12-Aug-80 08:43")
    (PACK* (Shorten x)
           TypeSeparator ext])

                                                                                                    9

(Shorten
  [LAMBDA (x)                                                (* R.Erickson "28-Aug-80 16:53")

          (* * Returns a copy of x, where all atoms have been shortened (reduced to that portion preceding the \, if any))

    (if (LISTP x)
        then (for el in x collect (Shorten el))
      elseif (LITATOM x)
        then (if ~x
                 then NIL
               elseif (GETHASH x ShortenHashArray)
               else (PUTHASH x (if (STRPOS TypeSeparator x)
                                   then (PACKC (for char in (DCHCON x ScratchList)
                                                  until char=(CONSTANT (CHCON1 TypeSeparator))
                                                  collect char))
                                        else x)
                                   ShortenHashArray))
      else x])

                                                                                                   10

(ShortenLessPrimes
  [LAMBDA (atom)                                             (* edited: " 6-Apr-79 08:52")
                                                             (* "(CHCON \) = 92,(CHCON ') = 39")
```

```
    (if (GETPROP atom 'ShortenLessPrimes)
      else (PUTPROP atom 'ShortenLessPrimes (PACKC (for x in (DCHCON atom ScratchList)
                                                     until x MEMB ShortenLessPrimesList
                                                     collect x])
)

(RPAQQ ForkFNS (CallSubsys FtpFile GtjfnHelp InitializeLongGtjfn NotifyMailer SetupXed TTYINFILE
                       TTYOUTFILE setBlockE stringPack))
(DEFINEQ
```

11

```
(CallSubsys
  [LAMBDA (Name Entry Acs File Subject)                    (* R.Bates "11-Jun-80 10:49")
    (PROG (error StartBlockE tempMod tempCoc1 tempCoc2 fork jfn name)
          (if (NLSETQ StartBlockE←(GETBLK 1))=NIL
              then (AffirmError "Can not find a free page" 'mild T)
                   (RETURN NIL))
          (for ac: in Acs do (setBlockE ac:1 ac:2) finally (if Subject~=NIL
                                                               then (stringPack Subject 7)))
          (if (OPENP File)=NIL
              then File←(OPENFILE File 'INPUT))
          (jfn←(OPNJFN File))
          (JSYS 18 (LOGOR -34359738368 jfn))
          (setBlockE 1 jfn)
          (tempMod←(JSYS 71 64 NIL NIL 2))
          (tempCoc1←(JSYS 74 64 NIL NIL 2))
          (tempCoc2←(JSYS 74 64 NIL NIL 3))
          (jfn←NIL)
          [error←(NLSETQ (PROGN fork←(SUBSYS Name NIL NIL 'DONTSTART)
                                name←(JSYS 127 NIL NIL NIL 1)
                                (ASSEMBLE NIL
                                          (MOVE 7 , CTCTP)
                                       T1 (HLRZ 1 , 0 (7))
                                          (TRNN 1 , 400000Q)
                                                          (* Skip if not assigned.)
                                          (JSYS 140Q)
```

(* JSYS DTI. Deassigns Terminal Interrupt code. Stepping thru the Lisp interrupt table pointed to by CTCTP (which stands for Channel Table ConTrol Pointer or something like that and just contains pointer to CHNTAB). Effect is to turn off all Lisp-activated interrupts I think.)

```
                                          (AOBJN 7 , T1))
                                (JSYS 112 fork (LOC StartBlockE))
                                                          (* SFACS 112)
                                (JSYS 129 fork Entry)     (* SFRKV 129)
                                (JSYS 115 fork)           (* WFORK 115)
                                (JSYS 113 fork (LOC StartBlockE))
                                                          (* 113 is RFACS)
                                (if Name='XED
                                    then jfn←(OPENR (LOC StartBlockE)+ 1))
                                (RELBLK StartBlockE 1)
                                (KFORK fork)
                                (JSYS 136 name)
                                (if (NUMBERP jfn)
                                    then (if jfn~=-1
                                             then jfn←(JFNS jfn)
                                                  (RLJFN -1)
                                                  (CLOSEF? File)
                                                  (OPENFILE jfn 'INPUT)
                                         else (RLJFN -1)
                                              (CLOSEF? File)
                                              (ERROR!))
                                  else (RLJFN -1)
                                       (CLOSEF? File]
          (ASSEMBLE NIL
                    (PUSHJ CP , SETINT)
                    (PUSHJ CP , SETMOD))
          (JSYS 143 65 tempMod)                           (* 143 STPAR)
          (JSYS 72 64 tempMod)                            (* 72 SFMOD)
          (JSYS 75 64 tempCoc1 tempCoc2)                  (* 75 SFCOC)
          (RETURN error])
```

12

```
(FtpFile
  [LAMBDA (File ForceFtp Compute HomeMachine ByteSize)     (* R.Bates " 6-Aug-80 10:28")
```

```
(* This function uses <LISPUSERS>FTP.COM. The function FTP moves things using the byte count and not by the page
count. Therefore if you move a file like a hash file which has bad byte count this function may move too much or too
little depending on the byte count.)


(PROG (free hostFile netFile)
      (if HomeMachine=NIL
          then HomeMachine←HOMEMACHINE)
      (if ~(NUMBERP ByteSize)
          then ByteSize←36)
      (if Compute=NIL
          then (if SYSTEMTYPE='TOPS20
                   then free←(JSYS 197 -1 NIL NIL 1) -(JSYS 197 -1 NIL NIL 2)
                                                          (* 197 is GTDAL)
                        (if free lt 250
                            then (if (MINUSP free)
                                     then (printout T .TAB0 0
                                                    "*** Directory is over allocated by "
                                                    (-free)
                                                    " pages" T)
                                   else (printout T .TAB0 0 "**** Directory has " free
                                                " pages left"
                                                T))
                                 (DISMISS 10000)))
                   free←(JSYS 140 (if SYSTEMTYPE='TOPS20
                                      then -17179688960
                                    else 262143)
                              NIL NIL 2)                  (* 140 is GDSKC)
                   (if free lt 2000
                       then (printout T .TAB0 0 "*** System has " free " pages left" T)
                            (DISMISS 10000)))
      (if (FILENAMEFIELD File 'DIRECTORY)=NIL
          then File←(if SYSTEMTYPE='TOPS20
                        then (PACK* '<(FILENAMEFIELD (DIRECTORYNAME T T)
                                                     'DIRECTORY)
                                    '> File)
                      else (PACK* '<(DIRECTORYNAME T T)
                                  '> File)))
      (if (for (i HELPFLAG) from 1 to 3 thereis (ERSETQ netFile←(FTP HomeMachine File 'INPUT
                                                                     'ANONYMOUS 'AFFIRMFTP NIL
                                                                     ByteSize)))=NIL
          then (printout T .TAB0 0 "Can not find file " File " on " HomeMachine T)
               (RETURN <'CanNotFindFile File>))
      (if ForceFtp
          then (if Compute
                   then (RETURN (CLOSEF netFile)))
               hostFile←(OPENFILE (PACKFILENAME (UNPACKFILENAME File))
                                  'OUTPUT NIL ByteSize)
        else (if (INFILEP hostFile←(PACKFILENAME < !(UNPACKFILENAME File)
                                                 'VERSION
                                                 (FILENAMEFIELD netFile 'VERSION)
                                                 >))
                 then (CLOSEF netFile)
                      (RETURN NIL)
               else (if Compute
                        then (RETURN (CLOSEF netFile)))
                    hostFile←(OPENFILE hostFile 'OUTPUT NIL ByteSize)))
      (printout T .TAB0 0 netFile " -> " hostFile T)
      (COPYBYTES netFile hostFile)
      (CLOSEF? netFile)
      (CLOSEF? hostFile)
      (RETURN netFile])
```

13

(GtjfnHelp

```
[LAMBDA (Message DontOpen File InputOrOuput)          (* R.Bates " 6-Aug-80 10:28")
```

(GtjfnHelp is used by TTYINFILE and TTYOUTFILE to get a file name from the user using the standard mointor call GTJFN. Message is a message to be printed each time the user is ask for a file name (can be NIL then a default value will be picked). The file the user gave is open and selected as the primary input or ouput file unless DontOpen is T. File if given implies a long gtjfn call (one where you can default fields). File can be list of the form UNPACKFILENAME gives or a LITATOM. (EXTENSION XGO) or .XGO would mean that the user wanted a file with extension xgo. ABORTED is return if the user hits control-E or an error occurs while this routine is running.)*

```
(* If NLSETQ fails then the jfn mode word (tempmod) and control character output control words
(tempcoc1 tempcoc2) are restored since GTJFN might have a chance to restore them.)
```

```
(PROG (temp tempMod tempCoc1 tempCoc2)
      (if File
          then                                          (* must be doing a long gtjfn call.)
               (if ~[PROG (HELPFLAG)
                          (RETURN (ERSETQ StartBlockE←(GETBLK 1]
                     then (AffirmError <"Can not get a new page." "Unable to do a long gtjfn call."
                                       >
                                       'mild)
                          File←NIL
                else (InitializeLongGtjfn File InputOrOuput)))
      (tempMod←(JSYS 71 64 NIL NIL 2))
      (tempCoc1←(JSYS 74 64 NIL NIL 2))
      (tempCoc2←(JSYS 74 64 NIL NIL 3))
      (temp←(NLSETQ (PROGN (do (if Message
                                   then (Dprintout T Message)
                                 else (Dprintout T T (if InputOrOuput='INPUT
                                                         then "Input file: "
                                                       else "Output file: ")))
                               (if (STREQUAL Message "")
                                   then Message←NIL)
                               [temp←(if File
                                         then (JSYS 16 (LOC StartBlockE))
                                       else                (* Do a GTJFN b2, b3, b4, b16 and b17 on.)
                                            (if InputOrOuput='INPUT
                                                then (JSYS 16 15033171968 16777281)
                                              else (JSYS 16 -27916500992 16777281]
                            repeatwhile (IGEQ temp 131072))
                           temp←(JFNS temp)
                           (RLJFN -1)
                           (if ~DontOpen
                               then (if InputOrOuput='INPUT
                                        then (INFILE temp)
                                      else (OUTFILE temp)))
                           temp)))
      (if File
          then (RELBLK StartBlockE 1))
      (JSYS 143 65 tempMod)                             (* 143 STPAR)
      (JSYS 72 64 tempMod)                              (* 72 SFMOD)
      (JSYS 75 64 tempCoc1 tempCoc2)                    (* 75 SFCOC)
      (if temp
          then (if (DRIBBLEFILE)
                   then                                 (* Fake a copy of what the user typed onto the DRIBBLE
                                                        file.)
                        (Dprintout NIL .TAB0 0 (if Message=NIL
                                                   then (if InputOrOuput='INPUT
                                                            then "Input file: "
                                                          else "Output file: ")
                                                 else Message)
                                               temp:1 " [confirm]" T))
               (RETURN temp:1)
        else (TERPRI)
             (RETURN 'ABORTED])
```

14

(InitializeLongGtjfn

```
[LAMBDA (File InputOutput)                              (* R.Bates "25-Feb-80 09:34")
                                                        (* Sets up a long GTJFN call by filling in appropriate
                                                        fields with Default values.)
    (PROG (temp version)
          (if (LITATOM File)
              then File←(UNPACKFILENAME File))
          (version←(if temp←('VERSION MEMB File) and temp←temp:2
                       then temp
                     else 0))
          (setBlockE 0 (if InputOutput='INPUT
                           then (LOGOR 15033171968 version)
                         elseif InputOutput='OUTPUT
                           then (LOGOR -27916500992 version)
                         else (AffirmError <"InputOutput not eq to INPUT or OUTPUT" InputOutput>
                                           'internal)))     (* e (0) flags)
          (setBlockE 1 16777281)                        (* e (1) I/O jfn)
          (for i from 2 to 6 as part in '(DEVICE DIRECTORY NAME EXTENSION PROTECTION)
             do (setBlockE i (if temp←(part MEMB File) and temp←temp:2
                                 then (stringPack temp (i-1)*10)
                                      (LOGOR -29947330560 (LOC StartBlockE)+(i-1)*10)
                               else 0)))
          (setBlockE 7 0)                               (* e (7) account)
          (setBlockE 8 0)                               (* e (10) JFN #)
```

```
    ])
```

15

```
(NotifyMailer
  [LAMBDA (DirectoryNumber)                                    (* R.Bates "22-Feb-80 13:31")
    (PROG (word wordNumber file ptr)
          (file←(OPENFILE (PACKFILENAME 'DEVICE (if SYSTEMTYPE='TENEX
                                                    then 'DSK
                                                  else 'PS)
                                        'DIRECTORY 'SYSTEM 'NAME 'MAILER 'EXTENSION 'FLAGS 'VERSION 1)
                          'BOTH 'OLD NIL '(WAIT THAWED)))
          (wordNumber←DirectoryNumber/36)
          (ptr←(MAPWORD wordNumber file))
          (word←(WORDCONTENTS ptr))
          [word←(LOGOR word (LLSH 1 35-(IREMAINDER DirectoryNumber 36]
          (SETWORDCONTENTS ptr word)
          (CLOSEF file])
```

16

```
(SetupXed
  [LAMBDA NIL                                                  (* R.Bates "26-Mar-79 15:16")
    (JSYS 72 262143 (LOGOR 258048 (JSYS 71 262143 NIL NIL 2)))
    (KFORK (SUBSYS '<SUBSYS>XED])
```

17

```
(TTYINFILE
  [LAMBDA (Message DontOpen File)                              (* R.Bates "25-Feb-80 09:11")
    (GtjfnHelp Message DontOpen File 'INPUT])
```

18

```
(TTYOUTFILE
  [LAMBDA (Message DontOpen File)                              (* R.Bates "25-Feb-80 09:06")
    (GtjfnHelp Message DontOpen File 'OUTPUT])
```

19

```
(setBlockE
  [LAMBDA (N X)
    (CLOSER (LOC StartBlockE)+N X])
```

20

```
(stringPack
  [LAMBDA (String Offset)                                      (* R.Bates "14-Mar-79 13:02")
                                                               (* Takes String and stores it the normal pdp-10 format
                                                               in the free page StartBlockE
                                                               (using setBlockE) starting at StartBlockE + Offset.)
    (for (char pos←4
               val←0)
       in < !(CHCON String)
            0>
       do (if (MINUSP pos)
              then pos←4
                   (setBlockE Offset val)
                   val←0
                   Offset←Offset+1)
          (val←(LOGOR (LLSH char (7*pos)+ 1)
                      val))
          (pos←pos-1)
       finally (setBlockE Offset val])
)

(RPAQQ HereForCompileFNS (DprintoutMacro EvalFormWithOutDribbleMacro Report TranslateApplyMacro
                                         UserProfileMacro))
(DEFINEQ
```

21

```
(DprintoutMacro
  [LAMBDA (X)                                                  (* R.Bates "13-Jun-80 13:07")
```

```
    <'PROG '(CurrentDribbleFile)
    <'if '(SETQ CurrentDribbleFile (DRIBBLEFILE))
      'then <'RESETLST '(RESETSAVE (SETDRIBBLEFILE))
              <'RESETSAVE '(OUTPUT CurrentDribbleFile) > <'printout ! X>>
      'else <'printout ! X>>>])
```

22

```
(EvalFormWithOutDribbleMacro
  [LAMBDA (X)                                          (* R.Bates "12-Jun-80 09:41")
    <'RESETFORM '(CloseAndOpenDribble T) ! X>])
```

23

```
(Report
  [NLAMBDA (Name%  Id%  Kind% )                        (* D.Thompson " 7-Nov-80 16:23")

        (* * All coded rewrite rules include a call to Report, just after the pattern test. If Report returns T, the rule is
        applied. This call is central to the manipulation of rewrite rules, particularly the reconstruction of rhs, and is
        controlled by global variables as follows:)


        (* * If REPORTFLAG and Kind MEMB RuleTypes, then the rule is printed (defns only if invoked) -
        UseRules must be true for any other activity to occur -
        If Kind = RuleTypes:Defn, Name must be in InvokedDefinitions to be rewritten. It will be removed from that list,
        unless ExpandAllDefns. -
        ListingFlag 1 (if = 'ZapRule, then ZapRule is called, now that we know id: flag is reset.) 2
        (if = 'IgnoreRule, nondefns whose lhs = RuleToBeIgnored are ignored. Used by SimplifyRule.) 3 (if otherwise true, will
        only allow the rule to be applied if args are symbolically identical to the LHS. This is used to extract a specific
        RHS. Sets KindOfRule, clears UseRules (which must thus be locally bound somewhere)))

                                                       (* list rule application. This is used for reporting,
                                                       also in discard.)
    [if REPORTFLAG and Kind%  MEMB RuleTypes and (ListingFlag='ZapRule
                                                    or (if Kind% =RuleTypes:Define
                                                           then Name%  MEMB InvokedDefinitions
                                                         else T))
        then (PROG (rule%  PS% )
                   (rule% ←(Rule (GetLHSide Name%  Id% )))
                   (PS% ←(for x%  in (PatternSubs rule% :LHS:Arguments (ARGLIST Name% ))
                             collect (create SubPair
                                             NEW ←(EVAL x% :NEW) using x% )))
                   (rule% ←(MakeSubs PS%  rule% ))
                   (printout NIL .TAB0 0 T (L-CASE Kind%  T)
                             , # (listOneRule rule% :LHS "->" rule% :RHS]
                                                       (* check for special opn codes)
    (if ListingFlag and UseRules
        then (if ListingFlag='ZapRule
                  then (ZapRule Name%  Id% )
                       ListingFlag←NIL
                elseif ListingFlag='IgnoreRule
                  then Kind% ~=RuleTypes:Define and ~(Name% =RuleToBeIgnored:LHS:Operator
                                                        and (EQUAL (GetLHSide Name%  Id% )
                                                                   RuleToBeIgnored:LHS))

                  else (if (EQUAL (STKARGS Name% )
                                  (GetLHSide Name%  Id% ):Arguments)
                           then                        (* only if exact match with LHS)
                                UseRules←NIL
                                KindOfRule←Kind%
                                T))
      else                                             (* vanilla)
           (UseRules and (if Kind% =RuleTypes:Define
                             then (if Name%  MEMB InvokedDefinitions
                                      then (if ExpandAllDefns
                                               else InvokedDefinitions←(REMOVE Name%  -
                                                                               InvokedDefinitions))
                                           T)
                           else T])
```

24

```
(TranslateApplyMacro
  [LAMBDA (X)                                          (* R.Bates "25-Feb-80 13:49")
    (PROG ((ApplyFun (X:1))
           (ArgFun (X:2))
           (Expr (X:3))
           (Arg1 (X:4))
```

```
          (NumberTimes (X:5)))
         (RETURN
           <'SELECTQ <'FLENGTH Expr>
             !(for i from 0 to (if NumberTimes
                                   else 8)
                 collect
                  <i
                    <'APPLY* ApplyFun Arg1
                      !(for j from 1 to i
                          collect
                           <(EVAL ArgFun)
                             (for k from 2 to j first $$VAL←Expr do $$VAL← <'CDR $$VAL>
                                finally $$VAL← <'CAR $$VAL>)
                             >)
                      >>
                 finally ($$VAL← < ! $$VAL <'APPLY ApplyFun
                                             <'CONS Arg1 <'for 'part 'in Expr 'collect
                                                           <(EVAL ArgFun)
                                                             'part >>>>>))
                   >])
```

25

```
(UserProfileMacro
  [LAMBDA (x)                                                    (* R.Bates "27-Sep-79 10:03")
    (if x::1=NIL or x:2=NIL
        then <'GETPROP (KWOTE 'UserProfile)
                (if (LISTP x:1) and x:1:1='QUOTE
                    then (KWOTE (U-CASE x:1:2))
                  else <'U-CASE x:1>)
                >
      elseif x:2=T
        then <'NOT <'FMEMB <'GETPROP (KWOTE 'UserProfile)
                             (if (LISTP x:1) and x:1:1='QUOTE
                                 then (KWOTE (U-CASE x:1:2))
                               else <'U-CASE x:1>)
                             >
                           ' OffValues >>
      else <'COND <x:2 <'NOT <'FMEMB <'GETPROP (KWOTE 'UserProfile)
                                       (if (LISTP x:1) and x:1:1='QUOTE
                                           then (KWOTE (U-CASE x:1:2))
                                         else <'U-CASE x:1>)
                                       >
                                     ' OffValues >>> <T <'GETPROP (KWOTE 'UserProfile)
                                                          (if (LISTP x:1) and x:1:1='QUOTE
                                                              then (KWOTE (U-CASE x:1:2))
                                                            else <'U-CASE x:1>)
                                                          >>>])
)

(RPAQQ InitFNS (InitXevalDtvs))
(DEFINEQ
```

26

```
(InitXevalDtvs
  [LAMBDA (MinfsTypeValue)                                       (* R.Bates "15-Feb-80 19:29")
    (XSet ALLOP ALL\Boolean)
    (XSet ANDOP AND\Boolean)
    (XSet EQVOP EQV\Boolean)
    (XSet IFOP IfThenElse)
    (XSet IMPOP IMP\Boolean)
    (XSet MAXOP MAX\Integer)
    (XSet MINOP MIN\Integer)
    (XSet NEOP NE\Boolean)
    (XSet NOTOP NOT\Boolean)
    (XSet OROP OR\Boolean)
    (XSet SOMEOP SOME\Boolean)
    (for operator in '(NEGOP MULTOP DIFFOP ADDOP LEOP QUOTIENTOP PWROP MODOP INVOP DIVOP LTOP GEOP
                         GTOP)
       as fun in '(MINUS\Integer TIMES\Integer DIFFERENCE\Integer PLUS\Integer LE\Integer
                                 QUOTIENT\Integer EXPT\Integer MOD\Integer INVERSE\Integer
                                 DIV\Integer LT\Integer GE\Integer GT\Integer)
       do (EVAL <'XSet operator fun>)
          [if (EXPRP fun)
              then (EDITE (GETD fun)
                          '(3 (MBD (BINXEVAL *]
       finally (DEFINE\OPR 'ADDOP ADDOP 'X\ADD\N 'NARGS 'FALSE)
               (DEFINE\OPR 'DIFFOP DIFFOP 'SIMP\DIFF 2 'TRUE)
```

```
                (DEFINE\OPR 'MULTOP MULTOP 'X\MULT\N 'NARGS 'FALSE)
                (DEFINE\OPR 'NEGOP NEGOP 'SIMP\NEG 1 'TRUE)
                (DEFINE\NARY\OP ADDOP 'N\ADD 0 'SIMP\NEG)
                (DEFINE\NARY\OP MINOP 'N\MIN POSINF 'NULL)
                (DEFINE\NARY\OP MULTOP 'NMULT 1 'MKINV)
                (DEFINE\NARY\OP MULTOP 'N\MULT 1 'NULL)
                (DEFINE\OPR 'LEOP LEOP 'SIMP\LE 2 'TRUE)
                (DEFINE\OPR 'LTOP LTOP 'SIMP\LT 2 'TRUE)
                (DEFINE\OPR 'GTOP GTOP 'SIMP\GT 2 'TRUE)
                (DEFINE\OPR 'GEOP GEOP 'SIMP\GE 2 'TRUE)
                (DEFINE\OPR 'QUOTIENTOP QUOTIENTOP 'MKQUOT 2 'TRUE)
                (DEFINE\OPR 'PWROP PWROP 'SIMP\POWER 2 'TRUE)
                (DEFINE\OPR 'MODOP MODOP 'SIMP\MOD 2 'TRUE)
                (DEFINE\OPR 'INVOP INVOP 'MKINV 1 'TRUE)
                (DEFINE\OPR 'DIVOP DIVOP 'SIMP\DIV 2 'TRUE))
    (#INIT\INFIX\PRINT)
    #LIST\OF\SYSTEM\OPS← < ! '(ExpressionWithType) ! #LIST\OF\SYSTEM\OPS> DTVSCANON←NIL
    (for typeValue in MinfsTypeValue do (MINFS typeValue:2 typeValue:1])
)

(RPAQQ ListFNS (/DREMASSOC ADDTOLIST AffirmSORT Closure Combinations DROPFROMLIST ElementsIn FoundIn
                           Minimize OccursAsOperatorIn POS REMOVEDUPLICATES Separate SubstWhenEq
                           UCI\SUBST UNMKLIST UPTO firstElement))
(DEFINEQ
```

27

(/DREMASSOC

```
  [LAMBDA (x lst)                                        (* R.Erickson "16-Aug-79 11:51")

          (* * destructively undoably remove all members of lst which start with x)


    (PROG (pre post)
          (while lst and lst:1:1=x do                    (* skip leading)
                                     (lst←lst::1))
          (pre←lst)
          (while pre do                                  (* maintain pre as a pointer to the last good cell in a
                                                            series. and post to the next good cell, if any)
                                                         (* assert pre:1 good)
                      (while pre and pre:2:1~=x do       (* 2 good. skip)
                                                   (pre←pre::1))
                                                         (* pre:1 good pre:2 bad)
                      (post←pre::1)                      (* bad)
                      (while post and post:1:1=x do      (* skip til good)
                                                    (post←post::1))
                      (/RPLACD pre post)
                      (pre←post))
          (RETURN lst])
```

28

(ADDTOLIST

```
  [LAMBDA (name element)                                 (* R.Erickson " 9-Oct-79 17:09")

          (* * like (push name element), but may be undone out of order, and evaluates args like /SET)


    (UNDOSAVE <'DROPFROMLIST name element >)
    (SET name <element !(EVALV name)
              >])
```

29

(AffirmSORT

```
  [LAMBDA (list method)                                  (* D.Thompson " 5-Nov-79 17:42")
    (if method='UPPER
        then (for x in (InterLispSort (for x in list collect <(U-CASE (MKLIST x):1) ! x>)
                                      T)
                    collect x::1)
      else (InterLispSort list method])
```

30

(Closure

```
  [LAMBDA (alist start)                                  (* R.Erickson "24-Oct-80 17:36")
```

```
(* * Transitive, nonReflexive closure. Returns all nodes reachable in one or more steps from start, using alist.
If start omitted, returns closure of whole alist.)


(if start
    then (for x in (ASSOC start alist)::1 join <x !(Closure alist x)
                                                >)
  else (for s in alist collect <s:1 !(REMOVEDUPLICATES (Closure alist s:1))
                               >])
```

31

```
(Combinations
  [LAMBDA (lists)                                        (* R.Erickson " 3-Sep-81 18:36")

(* * arg is a list of lists. Return a list of all tuples, where first element is from first list, etc)


  (if lists::1
      then                                               (* we vary the last elements first.
                                                         Inefficient, but pretty)
           (for one in lists:1 join (for comb in (Combinations lists::1) collect <one ! comb>))
    elseif lists
      then (for one in lists:1 collect <one>])
```

32

```
(DROPFROMLIST
  [LAMBDA (name element)                                 (* R.Erickson " 9-Oct-79 17:10")

(* * inverse of ADDTOLIST. Undoable out of order. Uses EQ test.)


  (UNDOSAVE <'ADDTOLIST name element>)
  (SET name (DREMOVE element (EVALV name])
```

33

```
(ElementsIn
  [LAMBDA (expression)                                   (* D.Thompson "25-Jun-80 10:16")
    (if (NLISTP expression)
        then (if expression
                 then 1
               else 0)
      else (for e in expression sum (ElementsIn e])
```

34

```
(FoundIn
  [LAMBDA (p q)                                          (* R.Erickson " 9-JAN-79 16:43")
    (OR (EQUAL p q)
        (AND (LISTP q)
             (for s in q thereis (FoundIn p s])
```

35

```
(Minimize
  [LAMBDA (list score)                                   (* R.Erickson "21-Sep-81 18:11")
                                                         (* list and function)

(* * Returns the first element of list such that (APPLY* score list) is a minimim. Score should be integer.)


  (PROG (cand (candScore MAX.INTEGER)
              sc)
        (for l in list do (sc←(APPLY* score l))
                          (if sc lt candScore
                              then candScore←sc
                                   cand←l))
        (RETURN cand])
```

36

```
(OccursAsOperatorIn
  [LAMBDA (atm ex)                                       (* D.Musser "14-May-79 16:15")
```

```
                                                    (* atm is an atom, ex is an expression.
                                                    Returns T if atm occurs in ex as an operator. NIL
                                                    otherwise)
    (if (LISTP ex)
        then atm=ex:Operator or (for arg in ex:Arguments thereis (OccursAsOperatorIn atm arg])
```

37

```
(POS
  [LAMBDA (key list)                                (* R.Erickson "20-Jul-79 19:11")
                                                    (* = 0 or n s.t. (NTH list n) :1 = key)
    (for l in list as i from 1 do (if l=key
                                      then (RETURN i))
       finally (RETURN 0])
```

38

```
(REMOVEDUPLICATES
  [LAMBDA (lst)
    (INTERSECTION lst lst])
```

39

```
(Separate
  [LAMBDA (list joinLists separator)                (* R.Erickson "29-Jan-81 16:22")

          (* * Place separators between the elements of list. If joinLists, we join LISTP elems after adding the commas.
          Note that we don't descent into elements.)


    (if ~separator
        then separator←Comma)
    (for tail on list join < !(if joinLists and (LISTP tail:1)
                                  then tail:1
                                else <tail:1>)
                           !(if tail::1
                                then <separator>
                                else NIL)
                           >])
```

40

```
(SubstWhenEq
  [LAMBDA (x y z)
    (if y=z
        then x
      elseif (LISTP z)
        then (for u in z collect (SubstWhenEq x y u))
      else z])
```

41

```
(UCI\SUBST
  [LAMBDA (X Y Z)                                   (* edited: " 6-Apr-79 09:26")
    (if (EQUAL Y Z)
        then X
      else (SUBST X Y Z])
```

42

```
(UNMKLIST
  [LAMBDA (lst)                                     (* R.Erickson "16-Aug-79 11:50")

          (* * opposite of MKLIST: to save space, we keep <atm> as atm)


    (if (NLISTP lst) or lst::1
        then lst
      else lst:1])
```

43

```
(UPTO
  [LAMBDA (x lst)                                   (* R.Erickson "24-Oct-80 17:46")
```

```
        (* * elements of lst upto. but not including, x. EQ test.)

    (for a in lst until a=x collect a])
```

44

```
(firstElement
  [LAMBDA (list)                                        (* D.Thompson " 4-Oct-79 16:11")
    (if (NLISTP list)
        then list
      else (firstElement list:1])
)

(RPAQQ PrintFNS (CloseAndOpenDribble AFFIRMMAPRINT BEEHIVE EnhanceHp Heading PRINTLINES
                                     EvalFormWithOutDribble))
(DEFINEQ
```

45

```
(CloseAndOpenDribble
  [LAMBDA (File)                                        (* R.Bates " 6-Aug-80 10:03")
    (PROG (temp HELPFLAG)
          (RETURN (if File=NIL
                      then NIL
                    elseif File=T
                      then temp←(DRIBBLE)
                           (if temp
                               then (printout T temp " is closed " T))
                           temp
                    else (CLOSEF? File)
                         temp←(ERSETQ (DRIBBLE File T T))
                         (if (NLISTP temp)
                             then (RecoverTranscript])
```

46

```
(AFFIRMMAPRINT
  [LAMBDA (header itemList lastSep tail)                (* D.Thompson " 5-Feb-80 14:57")

        (* * This routine prints a list in English format rather than parameter-style.)


    (PROG (lastOne rest)
          (if header
              then (printout NIL .TAB0 0)
                   (if header~=T
                       then (printout NIL header ,)))
          (if itemList~=NIL
              then (if (NLISTP itemList) or (EQLENGTH itemList 1)
                       then (printout NIL (MKLIST itemList):1)
                     else lastOne←itemList::-1
                          rest←(LDIFF itemList lastOne)
                          (MAPRINT rest NIL NIL (CONCAT (if (EQLENGTH itemList 2)
                                                            then NullString
                                                          else Comma)
                                                        (if lastSep~=NIL
                                                            then (if lastSep=T
                                                                     then " and"
                                                                   else (CONCAT Blank lastSep))
                                                          else NullString)
                                                        Blank lastOne:1)
                                                ", ")))
          (if tail
              then (printout NIL (if tail=T
                                     then Period
                                   else tail)
                             T))
          (printout NIL .TAB0 0])
```

47

```
(BEEHIVE
  [NLAMBDA (X)
    (PROG (bkspace)
          (if (L-CASE X)='on
```

```
            then bkspace←(PACKC '(4 32 4))
          else bkspace←(PACKC '(8 32 8)))
        (DELETECONTROL '1STCHDEL bkspace)
        (DELETECONTROL 'NTHCHDEL bkspace])
```

48

(EnhanceHp

```
  [LAMBDA (enhancement)                                    (* R.Erickson "15-Nov-79 16:29")

          (* * Enhancement is the letter for the display enhancement which should be sent to the terminal.)


    (RESETLST (RESETSAVE (ECHOCONTROL 27 'REAL)
                         '(PROGN (ECHOCONTROL 27 OLDVALUE)))
                                                           (* allow escapes)
              (Dprintout T Escape "&d" enhancement])
```

49

(Heading

```
  [LAMBDA (x)
    (TERPRI)
    (PRIN1 x)
    (TERPRI])
```

50

(PRINTLINES

```
  [NLAMBDA LINE

          (* PRINTLINES chunks down its parameter list using the following algorithm for printing: (1) print all strings as
          is, (2) if parameter is T then print a c.r., (3) otherwise EVAL the parameter and print its value if non-nil.)


    (for X in LINE bind VALUE do (if (STRINGP X)
                                     then (PRIN1 X)
                                   elseif X=T
                                     then (TERPRI)
                                   else (if VALUE←(EVAL X)
                                            then (PRIN1 VALUE)))
                                 (if $$LST1::1~=NIL and X~=T
                                     then (PRIN1 " "])
```

51

(EvalFormWithOutDribble

```
  [NLAMBDA X%                                              (* R.Bates " 1-Aug-80 16:09")
                                                           (* This function has a compiler macro.)
    (EVAL <'RESETFORM '(CloseAndOpenDribble T) ! X% >])
)

(RPAQQ XevalFNS (#INIT\XEVAL BINXEVAL DEFINE\NARY\OP DEFINE\OPR N2BINARY N2BINARY1 SIMPLIFY\BREAK
                             XSet))
(DEFINEQ
```

52

(#INIT\XEVAL

```
  [LAMBDA NIL                                              (* Edited by R.Bates on 13-JAN-78;
                                                           from version 62)
                                                           (* Note: ALPHA and DOMAIN are not currently being used
                                                           but were written by Don Good for some later use.)

          (* The "SIMP\Z" series of procedures make canonical expressions for operation Z from canonicalized arguments.
          N-ary operations have the suffix "\N" on their procedures. Procedures that XEVAL2 their own arguments have prefix "X\"

          instead of SIMP\. Procedures with prefix "N\" are the binary simplifying routines used by the N-ary simplifiers.
          SUPER\PUT\IN applies a binary simplifying routine for each N-ary operator (ADD, AND, MAX, MIN, MULT, OR) to each
          possible pair of arguments. The binary simplifying routines for N-ary operators (1) must be given
          (as the third arguement) the negative of the first argument. (2) do not receive arguemtns in order and must not
          reorder them. If any other routine needs to perform one of the N-ary operations, it must be accessed through
          DO\NARY\OP.)


    UNDEFINED←'UNDEFINED
```

```
    UFUNS←NIL
    STATE←NIL
    (DEFINE\OPR 'ADDOP 'PLUS 'X\ADD\N 'NARGS 'FALSE)
    (DEFINE\OPR 'ALLOP 'ALL 'SIMP\ALL 2 'TRUE)
    (DEFINE\OPR 'ANDOP 'AND 'X\AND\N 'NARGS 'FALSE)
    RECSETOP←'RecordWrite
    (DEFINE\OPR 'ASSIGNOP 'Assign 'SIMP\ASSIGN 3 'TRUE)
    (DEFINE\OPR 'RECACCESSOP 'RecordAccess 'SIMP\R\ACCESS 2 'TRUE)
    (DEFINE\OPR 'ASETOP 'ArrayWrite 'SIMP\A\SET 3 'TRUE)
    (DEFINE\OPR 'ASUBOP 'ArrayAccess 'SIMP\A\SUB 2 'TRUE)
    (DEFINE\OPR 'DIFFOP 'DIFFERENCE 'SIMP\DIFF 2 'TRUE)
    (DEFINE\OPR 'DIVOP 'DIV 'SIMP\DIV 2 'TRUE)
    (DEFINE\OPR 'EQOP 'EQ 'SIMP\EQ 2 'TRUE)
    (DEFINE\OPR 'EQVOP 'EQV 'SIMP\EQUIVALENT 2 'TRUE)
    (DEFINE\OPR 'GEOP 'GE 'SIMP\GE 2 'TRUE)
    (DEFINE\OPR 'GTOP 'GT 'SIMP\GT 2 'TRUE)
    (DEFINE\OPR 'IFOP 'IF 'X\IF 3 'FALSE)
    (DEFINE\OPR 'IMPOP 'IMP 'X\IMP 2 'FALSE)
    (DEFINE\OPR 'INVOP 'INVERSE 'MKINV 1 'TRUE)
    (DEFINE\OPR 'LEOP 'LE 'SIMP\LE 2 'TRUE)
    (DEFINE\OPR 'LTOP 'LT 'SIMP\LT 2 'TRUE)
    (DEFINE\OPR 'MAXOP 'MAX 'X\MAX\N 'NARGS 'FALSE)
    (DEFINE\OPR 'MINOP 'MIN 'X\MIN\N 'NARGS 'FALSE)
    (DEFINE\OPR 'MODOP 'MOD 'SIMP\MOD 2 'TRUE)
    (DEFINE\OPR 'MULTOP 'TIMES 'X\MULT\N 'NARGS 'FALSE)
    (DEFINE\OPR 'NEGOP 'MINUS 'SIMP\NEG 1 'TRUE)
    (DEFINE\OPR 'NEOP 'NE 'SIMP\NE 2 'TRUE)
    (DEFINE\OPR 'NOTOP 'NOT 'SIMP\NOT 1 'TRUE)
    (DEFINE\OPR 'OROP 'OR 'X\OR\N 'NARGS 'FALSE)
    (DEFINE\OPR 'PWROP 'EXPT 'SIMP\POWER 2 'TRUE)
    (DEFINE\OPR 'QUOTIENTOP 'QUOTIENT 'MKQUOT 2 'TRUE)
    (DEFINE\OPR 'SOMEOP 'SOME 'SIMP\SOME 2 'TRUE)
    (DEFINE\OPR 'SWAPOP 'ASWAP 'SIMP\SWAP 3 TRUE)
    (DEFINE\NARY\OP ADDOP 'N\ADD 0 'SIMP\NEG)
    (DEFINE\NARY\OP ANDOP 'N\AND TRUE 'SIMP\NOT)
    (DEFINE\NARY\OP MAXOP 'N\MAX NEGINF 'NULL)
    (DEFINE\NARY\OP MINOP 'N\MIN POSINF 'NULL)
    (DEFINE\NARY\OP MULTOP 'NMULT 1 'MKINV)
    (DEFINE\NARY\OP MULTOP 'N\MULT 1 'NULL)
    (DEFINE\NARY\OP OROP 'N\OR FALSE 'SIMP\NOT)
    XEVALTRACESET←NIL
    CONTEXT←TRUE])
```

53

```
(BINXEVAL
  [LAMBDA (X)                                    (* R.Bates "31-Oct-79 13:33")
    (N2BINARY (DeCanonicalize (XEVAL X])
```

54

```
(DEFINE\NARY\OP
  [LAMBDA (OPCODE BINARYFUN TOSSVALUE NEGFUN)    (* Edited by R.Bates on 14-JUL-77;
                                                 from version 24)
    (PUT OPCODE 'NARYS <BINARYFUN TOSSVALUE NEGFUN>])
```

55

```
(DEFINE\OPR
  [LAMBDA (OPNAME OPCODE EVALFUN ARGN ARGE)      (* Edited by R.Bates on 14-JUL-77;
                                                 from version 48)
                                                 (* Defines for each op: input name, proc, no of args.
                                                 and eval order)
    (SET OPNAME OPCODE)
    (PUT OPCODE 'EVFUN <EVALFUN ! ARGN>)
    (PUT OPCODE 'EVALARGS ARGE)
    (PUT OPNAME 'GLOBALVAR T])
```

56

```
(N2BINARY
  [LAMBDA (S)                                    (* edited: " 6-Apr-79 10:31")
    (if (ATOM S)
        then S
      elseif (MEMBER S:1 <ANDOP OROP ADDOP MULTOP MAXOP MINOP>)
        then (N2BINARY1 S:1 (for X in S::1 collect (N2BINARY X)))
      else <S:1 !(for X in S::1 collect (N2BINARY X))
```

```
          >])
```

57

```
(N2BINARY1
  [LAMBDA (OP LIS)

    (if LIS::2=NIL
        then <OP ! LIS>
      else <OP LIS:1 (N2BINARY1 OP LIS::1)
           >])
```

(Edited by R.Bates on 19-DEC-77; no file)*

58

```
(SIMPLIFY\BREAK
  [LAMBDA (PREFIX)
    NIL])
```

59

```
(XSet
  [NLAMBDA (X Y)
    (PUTPROP Y 'priority (GETPROP (EVAL X)
                                  'priority))
    (SET X Y])
)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 8b leftover utility functions - should be moved elsewhere)]

```
(RPAQQ UTILITYFNS (AFFIRMUSER AFFIRMWHEREIS AddToFile Affirm/ADDPROP BadFile CHRCT CallEDITF
                          CallEditor CheckLoad CloseDribble CorrectOneFileDate DWIMUSERFN Debug
                          EQCAR EXTENT FillOut GenIndex IsEq LexOrder MapExpr MatchArg MatchOp
                          PagesLeft PagesWarning RecoverTranscript TypeFile UpdateAffirm
                          baseNameSubst exec forget ruleSeqParse storage))
(DEFINEQ
```

60

```
(AFFIRMUSER
  [LAMBDA (wait default mess keylst typeahead lispxprntflg optionslst file dontCRafter)
                                                        (* D.Thompson "12-Feb-80 10:11")

        (* * This routine acts as the interface to the LISP routine ASKUSER. It determines if we can really ask the user a
        question, and does so if possible. If we can't ask, the default value is returned, if there is one.)


    (SELECTQ BatchMode
             [(T semi)
               (PROG (answer)                                  (* batch mode: don't ask, use default)
                     (RETURN (if BatchAnswers and answer←(validateValue NIL (PACK* (pop BatchAnswers)
                                                                                   Escape)
                                                                            NIL 'KEYLIST keylst NIL)
                                 then (printout NIL .TAB0 0 mess , "..." , answer T)
                                      answer
                                elseif default
                                  then

        (* * watch out for default = T, which means return value of NIL)

                                       (if default=T
                                           then default←NIL)
                                       (printout NIL .TAB0 0 mess , "..." ,
                                                 (if default
                                                   else NullString)
                                                 T)
                                       default
                                  else Prompt←BatchErrorPrompt
                                       (AffirmError "No default allowed: batch mode error."]
             [NIL                                           (* normal mode: ask the mess)
                  (if (FIXP wait)
                    else wait←NIL
                         default←NIL)
                  (if keylst
                    else keylst←'((Y "es")
                                  (N "o")))
                  (if 'CONFIRMFLG MEMB optionslst
                    else optionslst← < 'CONFIRMFLG T ! optionslst >)
                  (if 'PROMPTCONFIRMFLG MEMB optionslst
                    else optionslst← < 'PROMPTCONFIRMFLG T ! optionslst >)
                  (PROG1 (if typeahead
                             then (ASKUSER wait default mess keylst typeahead lispxprntflg optionslst
                                           file)
                           else (RESETBUFS (ASKUSER wait default mess keylst typeahead lispxprntflg
                                                    optionslst file)))
                         (if dontCRafter
                           else (printout NIL .TAB0 0]
             (Unexpected 'BatchMode T])
```

61

```
(AFFIRMWHEREIS
  [LAMBDA (NAME TYPE FILES FN)                          (* R.Bates "19-Mar-80 10:13")
    FILES←T                                             (* Always look threw the whereis hash file if possible.)
    (PROG (fileList)
          (fileList←(if (INFILEP WHEREIS.HASH) or (BadFile WHEREIS.HASH)
                        then (HashWHEREIS NAME TYPE FILES FN)
                      else FILES←FILELST                (* by doing this SETQ we disable looking threw the hash
                                                        file.)
                           (HashWHEREIS NAME TYPE FILES FN)))
          (if (LISTP (EVALV 'BADFILENAMES)) and NAME MEMB BADFILENAMES
              then fileList←(for file in fileList when file ~MEMB SYSTEMFILES collect file))

        (* BADFILENAMES is a list of functions, that the user request ownership of, and therefore the list of files returned
        by WHEREIS should not contain any system files on it. This will enable the user to pick where the new version of the
```

```
        function should go.)

        (for file in fileList do (CorrectOneFileDate file))
        (RETURN fileList])
```

62

```
(AddToFile
  [LAMBDA (item list kind)                                    (* R.Bates "26-Jun-80 21:33")
    list←(PACK* list kind)
    (/SET list <item !(if (EVALV list)='NOBIND
                          then NIL
                        else (EVALV list))
                >)
    (/SET list (INTERSECTION (EVAL list)
                             (EVAL list)))
    list])
```

63

```
(Affirm/ADDPROP
  [LAMBDA (ATM PROP NEW FLG)                                  (* R.Bates "22-Feb-80 08:45")
    (PROG (property)
          (RETURN (if property←(GETPROP ATM PROP)=NIL
                      then (/PUTPROP ATM PROP <NEW>)
                    elseif FLG
                      then (/PUTPROP ATM PROP <NEW ! property>)
                    else (/NCONC1 property NEW])
```

64

```
(BadFile
  [LAMBDA (File)                                              (* R.Bates " 4-Feb-80 13:26")
    (PROG (thisFile)
          (File←(UNPACKFILENAME File))
          [if thisFile←(INFILEP (PACKFILENAME File))
            else (for expr in SearchBadFile thereis thisFile←(INFILEP (PACKFILENAME
                                                                        <'DIRECTORY (EVAL expr)
                                                                          ! File>]
          (RETURN (if thisFile
                      then (LISPXPRIN1 '= T)
                           (RETURN (LISPXPRINT thisFile T])
```

65

```
(CHRCT
  [LAMBDA NIL #CURRENT\LINELENGTH-(POSITION])
```

66

```
(CallEDITF
  [LAMBDA (Name ListOfEditorComs)                             (* R.Erickson " 3-Jan-80 17:11")
    (if (Extension Name)=NIL
        then (PROG (msg)
                   (if Name=IFOP
                       then msg←(CONCAT IFOP " cannot be the main operator of the left hand side.")
                     else msg←(CONCAT Name " is not an user-defined operator," CR
                                      "  and therefore cannot be the main operator of the left hand side."))
                   (RETURN (AffirmError msg)))
      elseif ~((FNTYP Name)='EXPR or (FNTYP Name)='CEXPR)
        then (AffirmError (CONCAT Name " cannot be EDITED by the AFFIRM system." CR Name
                                  " is not an user-defined operator,"
                                  CR
                                  "  and therefore cannot be the main operator of the left hand side."))
      else (RESETVARS (INITIALS HOMEMACHINE)
                      (if (ERSETQ (APPLY 'EDITF <Name ! ListOfEditorComs>))
                        else (AffirmError <"Lisp editor returned error for" Name> 'internal])
```

67

```
(CallEditor
  [LAMBDA (Input Output Editor Question)                      (* R.Bates "12-Jun-80 09:08")
    (PROG (file outfile temp optionlist keylist error openFiles)
          (if Editor=NIL
```

```
        then Editor←(UserProfile 'TextEditor))
    (if (STRINGP Input) or (LISTP Input)
        then file←(OPENFILE 'TEMP.EDITOR 'OUTPUT)
             (if (LISTP Input)
                 then (PROG (FONTCHANGEFLG)
                            (printout file .PPV Input T))
               else (PROG (FONTCHANGEFLG)
                          (printout file Input T)))
      elseif (LITATOM Input)
        then (CLOSEF? Input)
             file←(if (INFILEP Input)
                      else (AffirmError (CONCAT "Can not find file: " Input)))
      else (AffirmError (CONCAT "Illegal agrument to CallEditor " Input)))
    (if Output=NIL
        then (if (STRINGP Input)
                 then Output←'STRING
               elseif (LISTP Input)
                 then Output←'LIST
               else Output←'FILE))
    (Editor←(U-CASE Editor))
    (if Editor MEMB '(XED)
        then Question←'NO)
    (openFiles←(OPENP))
    (if Editor ~MEMB '(XED SOS)
        then (CLOSEF? file)
             (printout NIL .TAB0 0 "The file to be edited is" , file T))
    (error←'NO)
    [EvalFormWithOutDribble (SELECTQ Editor
                                     (SOS error← (CallSubsys Editor 2 NIL file)
                                          (if error~=NIL
                                              then error←'NO))
                                     (XED error← (CallSubsys Editor 2 NIL file))
                                     (KFORK (SUBSYS Editor]
    (if error='NO
        then outfile←(UNPACKFILENAME file)
             temp←('VERSION MEMB outfile)
             temp:2←0
             outfile←[if (INFILEP (PACKFILENAME outfile))
                         else (AffirmError (CONCAT "Can not find file " (PACKFILENAME outfile]
             (OPENFILE outfile 'INPUT)
      elseif error=NIL
        then (RLJFN -1)
             (for file in (LDIFFERENCE (OPENP)
                                       openFiles)
                do (CLOSEF? file))
             (if ~(LITATOM Input)
                 then (DELFILE file))
             (RETURN 'ABORTED)
      else outfile←error:1)
    (if file~=outfile and ((STRINGP Input) or (LISTP Input))
        then (CLOSEF? file)
             (DELFILE file))
    (Output←(U-CASE Output))
    (if Output='FILE
        then temp←outfile
      elseif Output='LIST
        then temp←(if (STRINGP Input)
                      then (READFILE outfile)
                    else (READ outfile))
             (CLOSEF? outfile)
             (DELFILE outfile)
      elseif Output='STRING
        then (SETFILEPTR outfile 0)
             temp←""
             (while (GETFILEPTR outfile)
                    ~=(GETEOFPTR outfile) do temp←(CONCAT temp (READC outfile)))
             (CLOSEF? outfile)
             (DELFILE outfile)
      else (AffirmError (CONCAT "Illegal argument to CallEditor " Output)))
    (optionlist← NIL)
    (keylist← NIL)
    (if Question=NIL
        then Question←"Do you want to return the updated object? ")
    (if (U-CASE Question)
        ~='NO
        then (if (AFFIRMUSER NIL 'Y Question keylist T NIL optionlist)='N
                 then (if outfile~=Input
                          then (CLOSEF? outfile)
                               (DELFILE outfile))
                      (RETURN 'ABORTED)))
    (RETURN temp])
```

68

```
(CheckLoad
  [LAMBDA (type versions TypeName)                    (* R.Bates "11-Jan-80 12:30")

          (* * versions = (AFFIRMVERSION . AFFIRMSYSOUTFILE) The latter is the file for the most recent Sysout -- type = 'TYPE
          or 'PROOF)


    (if versions:1 lt 29
        then (printout T T

"**WARNING: This file was created before Affirm 29, so it is incompatible
with the new equality machanism. You should probably REREAD the axioms."
                    T))
    (if versions:1~=AFFIRMVERSION
        then (printout NIL T "(File created under Affirm" , versions:1 ")" T)
      elseif ~(EQUAL versions::1 AFFIRMSYSOUTFILE) and type~='TYPE
        then (printout NIL T "(File created in a different SYSOUT:" , versions::1 ")" T))
    (if type='TYPE
        then [if (NUMBERP versions:1) and (IGEQ versions:1 31)
                 then (if TypeName=NIL
                          then                        (* this is for backwards compatibility;
                                                      InitializeLoad has responsibility for this.
                                                      31 is when we changed SetupFile to insert call to
                                                      InitializeLoad in fileCOMS.)
                               (PROG (coms)
                                     [coms←(EVALV (FILECOMS (FILENAMEFIELD (INPUT)
                                                                           'NAME]
                                     (if coms~='NOBIND and coms and coms:-2:1='P
                                           and coms:-2:2:1='NoteDeclarations
                                        then           (* extract the proper spelling of the type name from the
                                                       filecoms list)
                                             TypeName←coms:-2:2:2:2
                                       elseif coms~='NOBIND and coms:-1:1='P
                                                 and coms:-1:2:1='InitializeLoad
                                        then TypeName←coms:-1:2:3
                                                       (* version # is bad, but this is a recent file)

                                        else           (* can't determine the type name;
                                                       punt)
                                             (AffirmError <"** this file " (INPUT)

                                                          " has strange format; see Affirm personnel"
                                                          >]
             (if TypeName~=NIL
                 then (DeclareType TypeName T])
```

69

```
(CloseDribble
  [LAMBDA NIL                                         (* R.Bates "11-Aug-80 20:19")
    (PROG (file)
          (if file←(DRIBBLEFILE)=NIL
              then (printout T "No transcript file open" T)
                                                      (* file T so can do during compiles,etc.)
           else (EvalFormWithOutDribble (CallEditor file 'File NIL 'NO))
                (RETURN file])
```

70

```
(CorrectOneFileDate
  [LAMBDA (File)                                      (* R.Bates " 5-Feb-80 11:27")
    (PROG (newFile oldFile property)
          (if ~(property←(GETPROP File 'FILEDATES))
              then (RETURN NIL))
          (property←property:1)
          (if (INFILEP oldFile←property::1)
              then (RETURN <oldFile>))
          (if newFile←(BadFile oldFile)
              then property::1←newFile
                   (RETURN <newFile>])
```

71

(DWIMUSERFN

```
[LAMBDA NIL                                                    (* R.Erickson " 3-Aug-81 18:47")
  (PROG ((faultx FAULTX)
         (faultargs FAULTARGS)
         (faultapplyflg FAULTAPPLYFLG)
         ownerType)
        (if (ATOM faultx) and ownerType←(IsType (Extension faultx))
            then (if faultapplyflg
                     then                                      (* nilary Skolem function)
                          (MakeFunction <faultx ! faultargs>)
                          (RETURN faultx)
                   elseif (FASSOC (ShortenLessPrimes faultx)
                                  ownerType:LocalDeclarations)
                     then

(* a declared variable which the user entered in a primed form not yet generated by NewSymbolFrom.
We check that it corresponds to its type. Note that ownerType might not be CurrentType; if user loaded a file,
schemas won't be evaluated util printing.)


                          (/SET faultx faultx)
                          (RETURN faultx))
          elseif (LISTP faultx) and (IsType (Extension faultx:Operator))
            then                                               (* internal Skolem fn with args)
                 (RETURN (KWOTE (MakeFunction faultx])
```

72

```
(Debug
[LAMBDA (Off)                                                  (* R.Bates "23-Jan-80 15:56")
  (if (NLISTP (EVALV 'InitialMinfs))
      then InitialMinfs←(for (i j) from 1 to 30 collect <i ! j> when j←(MINFS NIL i)))
  (for j in (if Off
                then InitialMinfs
              else DebugMinfs)
     do (MINFS j::1 j:1))
  (if ~Off
      then (for file in FILELST do (REMPROP file 'DATABASE) when (GETPROP file 'DATABASE)='NO)
           SAVEDBFLG←'ASK
           LOADDBFLG←'ASK
    else SAVEDBFLG←'NO
         LOADDBFLG←'NO])
```

73

```
(EQCAR
[LAMBDA (U V)
  (AND ~(ATOM U)
       U:1=V])
```

74

```
(EXTENT
[LAMBDA (x)                                                    (* D.Musser "16-OCT-78 11:56")
  (if (LISTP x)
      then (for y in x sum (EXTENT y))
    else 1])
```

75

```
(FillOut
[LAMBDA (list length fillElement)                              (* D.Thompson "15-Oct-79 09:21")
  (PROG (#needed)
        (length←(if (FIXP length)
                    then length
                  elseif (LISTP length)
                    then (FLENGTH length)
                  elseif length=NIL
                    then 0
                  else 1))
        (RETURN (if (ILEQ length 0)
                    then list
                  else list←(MKLIST list)
                       (if #needed←length-(FLENGTH list) gt 0
                           then < ! list !(for i from 1 to #needed collect fillElement) >
                         else list])
```

76

```
(GenIndex
  [LAMBDA (OutFile FileList)                                    (* R.Bates "13-Dec-79 09:19")
    (PROG (duplicates fileNames fullSystem functions positions)
          (if OutFile=NIL
              then OutFile←T)
          (if FileList=NIL
              then FileList←SYSTEMFILES
                   fullSystem←T
                   fileNames←(SORT (for X in FULLSYSTEMFILES collect X::1))
            else fileNames←FileList)
          (if OutFile~=T
              then OutFile←(OPENFILE OutFile 'OUTPUT))
          (positions←(SORT (for file in FileList join (for fn in (PROGN (LOADFROM file)
                                                                        (FILEFNSLST file))
                                                                 as i from 1 collect <(U-CASE fn)
                                                                                      fn file i>))
                           T))
          (if fullSystem
              then (printout OutFile .SKIP 3 10 "This index is generated from the system "
                             AFFIRMMAKESYSFILE 10 " using these files:")
            else (printout OutFile .SKIP 3 10 "This index is generated from these files:"))
          (printout OutFile .PARA 15 0 fileNames .SKIP 5)
          (for entry in positions as i from 1
             do (printout OutFile 10 .I5 i 20 entry:2 55 entry:3 75 .I3 entry:4 T))
          (functions←(for entry in positions collect entry:2))
          (duplicates←(for fn on functions by fn::1 bind f eachtime f←fn:1 collect f
                         when f FMEMB fn::1))
          (if duplicates
              then (first (printout OutFile .PAGE 10
                                    "The following functions have duplicate entries:"
                                    T)
                          for (fn FONTCHANGEFLG) in duplicates do (printout OutFile 10 fn 35 .PPV
                                                                            (WHEREIS fn)
                                                                            T)))
          (RETURN (CLOSEF OutFile])
```

77

```
(IsEq
  [LAMBDA (op)                                                  (* D.Musser "10-Aug-79 14:54")
                                                                (* op is an atom. Returns T if op is an equality
                                                                operator, NIL otherwise.)
    (OR op=EQOP op:EQOP])
```

78

```
(LexOrder
  [LAMBDA (x y)                                                 (* edited: " 6-Apr-79 09:26")
                                                                (* Predicate which is T if x is lexicographically
                                                                strictly less than y, NIL otherwise)
    (if (LISTP x)
        then (if (LISTP y)
                 then [for a on x as b on y always (EQUAL a:1 b:1)
                         finally (RETURN (if a
                                             then (if b
                                                      then (LexOrder a:1 b:1)
                                                    else NIL)
                                           else (if b
                                                    then T
                                                  else NIL]
                 else NIL)
      elseif (LISTP y)
        then T
      elseif (EQUAL x y)
        then NIL
      else (ALPHORDER x y])
```

79

```
(MapExpr
  [LAMBDA (ex fn onlyLists)                                     (* R.Erickson "15-Sep-80 18:55")
          (* * applies fn to all subexpressions of ex, including top. if onlyLists. doesn't do so for atoms)
    (if (LISTP ex)
```

```
      then (APPLY* fn ex)
           (for el in ex do (MapExpr el fn onlyLists))
    elseif onlyLists
    else (APPLY* fn ex])
```

80

```
(MatchArg
  [LAMBDA (Expr Pattern)                                  (* R.Bates "18-Sep-80 12:56")
    (if (LISTP Expr) and (LISTP Pattern)
        then (MatchOp Expr Pattern)
      elseif (Shorten Expr)=Pattern
        then T
      elseif (LISTP Pattern)
        then NIL
      elseif (LISTP Expr)
        then (Expr:1 MEMB SkolemFunctions) and (MatchArg (Shorten Expr:Operator)
                                                         Pattern)
      else NIL])
```

81

```
(MatchOp
  [LAMBDA (Expr Pattern)                                  (* R.Bates "18-Sep-80 09:28")
    (if (NLISTP Expr) or (NLISTP Pattern)
        then (MatchArg Expr Pattern)
      elseif (LENGTH Expr)
             ~=(LENGTH Pattern)
        then NIL
      elseif (Shorten Expr:Operator)
             ~=Pattern:Operator
        then NIL
      else (for x in Expr as pat in Pattern always (MatchOp x pat])
```

82

```
(PagesLeft
  [LAMBDA NIL                                             (* D.Thompson "29-Aug-80 12:25")

        (* * This routine performs the magickal incantations necessary to determine the number of core pages still
        unallocated to specific data types by the Interlisp garbage collector.)


    (for i from 0 to 510 count (CAR (VAG (LOC (COREVAL TYPTAB))+i))=(VAG 0])
```

83

```
(PagesWarning
  [LAMBDA (pages)                                         (* D.Thompson "15-Oct-80 12:56")

        (* * This routine warns the user to BEWARE! when the number of free pages drops below its argument
        (default is 10, quite arbitrarily).)


    (PROG (p)
          (if (FIXP pages)
            else pages←10)
          (if FewPagesMessage
            elseif (ILEQ p←(PagesLeft)
                         pages)
              then (printout NIL .TAB0 0 T "*** WARNING: You only have" , p , "free pages left." T)
                   (if (ILEQ p 10)
                       then (printout NIL .TAB0 13 "Better start saving what you can!" T 13
                                      "Then plan on restarting in a fresh AFFIRM."   -
                                      T))
                   (if 'None ~=(UserProfile 'GarbageCollectionMessage)
                       then (if (ILEQ (UserProfile 'GarbageCollectionPages)
                                      p)
                                then (UserProfileSet 'GarbageCollectionPages p))
                            (UserProfileSet 'GarbageCollectionMessage 'Compact))
                   FewPagesMessage←T])
```

84

```
(RecoverTranscript
  [LAMBDA NIL                                             (* R.Bates "15-Aug-80 13:52")
```

```
(PROG (HELPFLAG)
      (if TRANSCRIPTFILE=NIL
          then (RETURN))
      (if TRANSCRIPTFILE~='GiveMessage
          then (if (OPENP TRANSCRIPTFILE) and (ERSETQ (SETDRIBBLEFILE TRANSCRIPTFILE))
                                              ~=NIL
                   then (CloseAnnotation)
                        (Dprintout NIL .TAB0 0 (if Annotating
                                                   then "@Comment["
                                                  else "")
                                               "Lost transcript recovered."
                                               (if Annotating
                                                   then "]"
                                                 else "")
                                               T)
                        (RETURN))
               (if (INFILEP TRANSCRIPTFILE) and (ERSETQ (PROGN (CLOSEF? TRANSCRIPTFILE)
                                                               (DRIBBLE TRANSCRIPTFILE T T)))
                                                    ~=NIL
                   then (CloseAnnotation)
                        (Dprintout NIL .TAB0 0 (if Annotating
                                                   then "@Comment["
                                                  else "")
                                               "Lost transcript recovered."
                                               (if Annotating
                                                   then "]"
                                                 else "")
                                               T)
                        (RETURN)))
      (TRANSCRIPTFILE←NIL)
      (printout NIL .TAB0 0 T # (PRINTBELLS)
                "You don't have any transcript file!" T])
```

85

(TypeFile

```
[LAMBDA (fileName)                                        (* D.Thompson " 5-Sep-80 20:08")
```

(* This routine copies the contents of the file whose name is provided as the parameter, to the primary output file (usually the terminal).)*

```
(if TYPEFILENAME←(getFileName fileName NIL "File to copy: ")
    then (if (OPENP TYPEFILENAME 'INPUT)
             then TYPEFILEPOINT←(GETFILEPTR TYPEFILENAME)
           else TYPEFILEPOINT←NIL
                (OPENFILE TYPEFILENAME 'INPUT))
         (printout NIL .TAB0 0 TYPEFILENAME ":" T)
         (RESETLST (RESETSAVE NIL '(PROGN (COND
                                            ((NUMBERP TYPEFILEPOINT)
                                              [COND
                                                ((NOT (OPENP TYPEFILENAME (QUOTE INPUT)))
                                                  (OPENFILE TYPEFILENAME (QUOTE INPUT]
                                              (SETFILEPTR TYPEFILENAME TYPEFILEPOINT))
                                            (T (CLOSEF? TYPEFILENAME)))
                                          (SETQ TYPEFILEPOINT NIL)))
                   (COPYBYTES TYPEFILENAME))
  else (AffirmError])
```

86

(UpdateAffirm

```
[LAMBDA (ListOfFiles SystemUpdate Compute HomeMachine)    (* R.Bates "24-Jan-80 13:10")
  (PROG (temp)
        (LOAD? (PACKFILENAME 'DIRECTORY 'LISPUSERS 'NAME 'NET 'EXTENSION 'COM))
        (LOAD? (PACKFILENAME 'DIRECTORY 'LISPUSERS 'NAME 'PASSWORDS 'EXTENSION 'COM))
        (LOAD? (PACKFILENAME 'DIRECTORY 'LISPUSERS 'NAME 'FTP 'EXTENSION 'COM))
        (if SystemUpdate
            then (if SystemUpdate='BASIS
                     then ListOfFiles← < !(for x in PV\BASIS\FILES collect (FASSOC (U-CASE x::1)
                                                                                  FULLSYSTEMFILES))
                                         ! ListOfSystemFiles>
                   else ListOfFiles← < ! FULLSYSTEMFILES ! FilesToUpdate ! ListOfFiles>))
        (RETURN (REMOVE NIL (for file in ListOfFiles
                               collect (if (LISTP file)
                                           then file←file::1)
                                       (if (FILENAMEFIELD file 'VERSION)
                                           then (if ~(INFILEP (PACKFILENAME (UNPACKFILENAME file)))
                                                    then (FtpFile file T Compute HomeMachine))
```

```
                    else (FtpFile file NIL Compute HomeMachine])
```

87

```
(baseNameSubst
  [LAMBDA (variable expression)                    (* D.Musser "11-Jul-79 15:42")
                                                   (* Substitutes expression for the "base name" of
                                                   variable)
    (if (ATOM variable)
        then expression
      else (SELECTQ (Shorten variable:SYNTACTICTYPE)
                    (ArrayAccess (create ArrayAccess
                                         Array ←(baseNameSubst variable:Array expression)
                                 using variable))
                    (RecordAccess (create RecordAccess
                                          Record ←(baseNameSubst variable:Record expression)
                                  using variable))
                    (HeapOrFileAccess (create HeapOrFileAccess
                                              HeapOrFile ←(baseNameSubst variable:HeapOrFile
                                                                         expression)
                                      using variable))
                    (PRINTLINES T "*** Unrecognized variable" variable T])
```

88

```
(exec
  [LAMBDA NIL                                      (* D.Thompson "10-Sep-80 14:51")

        (* * This routine starts up the exec as a lower fork. -
        An AFFIRM command function.)


    (printout NIL .TAB0 0 "Type" , (if (SYSTEMTYPE)='TENEX
                                       then "QUIT"
                                     else "POP")
              , "to return to AFFIRM." T)
    (OR [PROG (HELPFLAG)
              (RETURN (NLSETQ LASTEXEC←(SUBSYS LASTEXEC]
        [PROG (HELPFLAG)
              (RETURN (NLSETQ LASTEXEC←(SUBSYS]
        (AffirmError "Can't get to the exec -- unable to create a lower fork."])
```

89

```
(forget
  [LAMBDA NIL                                      (* D.Thompson "31-Oct-79 15:31")

        (* * This function performs the LISP command FORGET. -
        An AFFIRM command function.)


    (for X in LISPXHISTORY:1 do (UNDOLISPX2 X T])
```

90

```
(ruleSeqParse
  [LAMBDA (ruleSeq kind)                           (* R.Erickson "10-Jan-80 12:45")

        (* * Called by ProcessAFFIRMCommand, processes equational rules from the parser and checks their form.
        Calls the proper rule-creating function with the raw untranslated equation)


    (for rulerec in ruleSeq:rule bind expr
       do                                          (* compute the proper expression form)
        (expr←rulerec:expression)
        (if rulerec:expression#
            then expr← <EQOP expr rulerec:expression#>
          elseif expr:Operator=EQOP
          elseif kind MEMB '(axiom lemma) and (Translate expr):Type='Boolean
            then expr← <IFOP expr TRUE FALSE>
          else (printout NIL .TAB0 0 "The formula" , # (PrettyPrint expr T)
                         T "isn't an equation." T)
               (AffirmError))
        (SELECTQ kind
                 ((axiom lemma)
                   (Axiom expr kind))
                 (schema (Schema expr))
```

```
                (define (Define expr))
                (AffirmError "Unexpected kind in ruleSeqParse" 'internal])
```

91

```
(storage
  [LAMBDA (state)                                        (* R.Bates "23-Jan-80 13:40")

          (* * This routine modifies storage parameters (useful when storage gets tight). -
          An AFFIRM command function.)


    (PROG [oldSlice state# stg slice (states ('(normal tight severe]
          (if state
            else state←'normal)
          (state#←(FLENGTH (state MEMB states)))          (* (1 severe) (2 tight) (3 normal))
          (if state#=0
              then (AFFIRMMAPRINT "The storage parameter should be one of" states "or" T)
                   (AffirmError))
          (for level in MinfsLevels do                    (* LISP wont set levels below 25)
                                       (stg←(FNTH level::1 state#):1)
                                       (if stg~=(MINFS NIL level:1)
                                           then (MINFS stg level:1)))
          (slice←(CAR (FNTH '(3 5 30)
                            state#)))
          (oldSlice←(UserProfile 'HistoryWindowSize))
          (if slice~=oldSlice
              then (if (UserProfileSet 'HistoryWindowSize slice T)
                       then (printout NIL .TAB0 0 "Changed history window size from" , oldSlice ,
                                      "to"
                                      ,
                                      (UserProfile 'HistoryWindowSize)
                                      "." T])
)

(RPAQ AFFIRMVERSION 0)

(RPAQQ SearchBadFile ((DIRECTORYNAME)
                      PV\DIRECTORY
                      (QUOTE SS:)
                      (PACK* (QUOTE SS:)
                             (QUOTE <)
                             PV\DIRECTORY
                             (QUOTE >))))

(RPAQQ TEMPRELEASECOMMENTS <AFFIRM>RELEASEDATAFILE.COMMENTS.1)

(RPAQQ VERSIONFILE <AFFIRM>VERSION.AFFIRM)

(RPAQ AFFIRMINITFIRSTTIME NIL)

(RPAQQ DebugMinfs ((1 . 10000)
                   (4 . 512)
                   (5 . 512)
                   (8 . 10000)
                   (9 . 512)
                   (12 . 1000)
                   (16 . 512)
                   (18 . 3000)
                   (24 . 512)
                   (28 . 512)
                   (30 . 512)))

(RPAQ ExtensionHashArray < (HARRAY 200)
                         ! 1.5>)

(RPAQQ MinfsLevels ((ARRAYP 25 512 1500)
                    (LISTP 512 5000 10000)
                    (LITATOM 25 200 1000)
                    (FLOATP 25 512 512)
                    (FIXP 25 200 1000)
                    (ATOM.CHARS 25 512 512)))

(RPAQ ScratchList (for I from 1 to 35 collect NIL))

(RPAQ ShortenHashArray SYSHASHARRAY)

(RPAQQ SYSTEMSTATECOMS [(COMS * (LIST (CONS (QUOTE VARS)
                                            Switches)
                                      (CONS (QUOTE HORRIBLEVARS)
```

```
PROOFVARS])

(/MOVD (QUOTE Affirm/ADDPROP)
       (QUOTE /ADDPROP))
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* LIMBO: apparently useless)]

```
(RPAQ any 'any)

(RPAQ IMPLIST NIL)

(RPAQ Prompt "U: ")
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 9 Error handling, interrupts)]

```
(RPAQQ AFFIRMErrorFNS (AffirmBacktrace AffirmBreak AffirmError ParsingError Unexpected))
(DEFINEQ
```

92

```
(AffirmBacktrace
  [LAMBDA (whence)                                        (* D.Thompson "22-Nov-79 19:16")

        (* * print a backtrace, stopping at a ProcessCommand or a depth of 15 calls.)


    (PROG (startpos endpos (Counter 0)
                (Limit (SELECTQ whence
                               (break 5)
                               15)))
          (DECLARE: (SPECVARS Counter Limit))
          (startpos←(SELECTQ whence
                         (break (STKNTH 1 'BREAK1))
                         (AffirmError (STKNTH 1 'AffirmError))
                         (REALSTKNTH 3)))                 (* bottom of backtrace)
          (endpos←((CDR (SEARCHPDL (FUNCTION [LAMBDA (name frame)
                                       (if (name='ProcessCommand) or Counter gt Limit
                                           then                (* return value- stop at this frame.
                                                               Searchpdl returns (name . stackptr))
                                                T
                                             elseif (REALFRAMEP frame)
                                               then Counter←Counter+1
                                                    NIL])
                                   startpos))
               or T))
          (BAKTRACE startpos (REALSTKNTH 1 endpos)
                    NIL 13)                                (* arguments, locals; no blips, temporaries.
                                                           BAKTRACE, as opposed to BACKTRACE, does not print ending
                                                           position; hence REALSTKNTH up by one.)
      ])
```

93

```
(AffirmBreak
  [LAMBDA (direction)                                     (* R.Erickson "22-Sep-80 15:18")

        (* * This gets called by an entry on BREAKRESETFORMS, and serves to protect naive users from Interlist breaks.
        Return to caller if break is allowed, else print a message and ERROR!. We don't worry about ↑B or ↑H, since they are
        disabled when appropriate.)


        (* * The Specvar BRKTYPE will now be INTERRUPT for ↑H, ERRORX for errors, NIL for a requested break, and
        (undocumented!) an array for ↑B. For Trace, BRKCOMS:1 is GO or TRACE. Unfortunately, a NIL BRKTYPE will be called
        for both break entry and for an OK, so we can't comfortably allow requested breaks to proceed.)


    (SELECTQ direction
            (LEAVING)
            [ENTERING (PROG (permit)
                            (permit←(if (FGETD 'UserProfile)
                                        then (UserProfile 'BreakAccess T)
                                      else T))
                            (if permit or BRKCOMS:1 MEMB '(TRACE GO) or ~BRKTYPE
                                then                       (* allow the break or trace to proceed)
                                     (RETURN 'LEAVING)
                              else (printout NIL .TAB0 0 "Suppressing Interlisp break on" , .PPVTL
                                             BRKFN T "Internal Diagnostic Trace:")
                                   (AffirmBacktrace 'break)
                                   (ERROR!]
            (SHOULDNT])
```

94

```
(AffirmError
  [LAMBDA (MESS kind quietflg)                            (* R.Bates " 6-Aug-80 10:28")
                                                          (* kind is one of (mild normal internal) quietflg
                                                          suppresses kind-specific notes to user)

    (if ~(MESS or kind)
        then quietflg←T)
```

```
    kind←(L-CASE kind)
    (if kind='internal
        then (printout NIL .TAB0 0 "AFFIRM internal error:" T))
    (printout NIL .TAB0 0 .PARA 0 0 (MKLIST MESS)
              T)
    (SELECTQ kind
             (mild                                          (* return to caller)
                   NIL)
             [internal (if (UserProfile 'BreakAccess T)
                           then (RESETVARS ((HELPFLAG 0))   (* force a break)
                                           (ERROR "Entering Interlisp Break"))
                       else (printout NIL .TAB0 0 "Internal Diagnostic Trace:" #
                                      (AffirmBacktrace 'AffirmError)
                                      .TAB0 0 T "AFFIRM may be in an unsound state." T)
                            (if ~quietflg
                                then (ERROR!]
             ((nil normal)
               (ERROR!))
             (AffirmError <"Illegal arg to AffirmError" kind> 'internal])
```

95

(ParsingError

```
  [LAMBDA (dontReturn)                                  (* D.Thompson "26-Feb-80 11:43")

          (* * This routine prints the syntax error message. and a certain amount of the sequence of input tokens.)


    (PROG (last)
          (last←(if last←(LASTN PARSEDTOKENS (UserProfile 'ErrorTokensOutput))
                    then <'... ! last::1>
                  else PARSEDTOKENS))
          (printout NIL .TAB0 0 "Error in parsing:" T .PPVTL last T)
          (if dontReturn
              then (AffirmError])
```

96

(Unexpected

```
  [LAMBDA (category functionName parameters)            (* D.Thompson "25-Oct-79 12:51")

          (* * This routine is expected to return NIL)


    (PROG (done #extra formals newElement)
          (category←(U-CASE category))
          (parameters←(MKLIST parameters))
          (newElement← <functionName parameters>)
          [for u in Unexpected until done
             do (if category=u:Category
                    then done←T
                         (if (MEMBER newElement u:Elements)
                           else u:Elements← < ! u:Elements newElement>
                                (printout NIL T "Unexpected" , u:Category , "found in routine" ,
                                          functionName "." T)
                                (if (LITATOM functionName) and (FGETD functionName)
                                    then formals←(ARGLIST functionName)
                                  else formals←NIL
                                       (printout NIL "(undefined routine!)" T))
                                (printout NIL "Parameters:" ,)
                                (if formals~=NIL or parameters~=NIL
                                    then (printout NIL .PPVTL (for formal
                                                                 in (FillOut formals parameters
                                                                            '!extra!)
                                                                 as actual
                                                                 in (FillOut parameters formals NIL)
                                                                 collect <formal '= actual>)
                                                          T)
                                  else (printout NIL "(none)" T]
          (if done
            else (Unexpected 'UnexpectedCategory 'Unexpected <category functionName parameters>))
          (RETURN NIL])
)

(RPAQ AskUserToDefine T)

(RPAQ BADFILENAMES NIL)

(RPAQ DTVSCOMPLAIN T)
```

```
(RPAQ DWIMUSERFN T)

(RPAQQ HELPDEPTH 4)

(RPAQQ NOSAVESETVARS (ZapKind KnownTypes))

(RPAQQ UnexpectedCategories (AFFIRMObject File HelpTopic LineDelete Need PrintObject ProfileEntryName
                                    UnexpectedCategory))

(RPAQ Unexpected (for category in UnexpectedCategories collect (create UnexpectedList Category ←
                                                                       (U-CASE category)
                                                                       Elements ← NIL)))

(RPAQ UserInterrupt NIL)

(ADDTOVAR ERRORTYPELST (23 (BadFile (CADR ERRORMESS)))
                       (42 (BadFile (CADR ERRORMESS))))

(ADDTOVAR BREAKRESETFORMS (AffirmBreak (QUOTE ENTERING)))
(INTERRUPTCHAR 24 'UserInterrupt)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 10a System gens, sysouts, files, user initials)]

```
(RPAQQ SYSTEM\MAKERFNS (AffirmInitAfterMakesys AffirmMakesys AffirmSysoutInit InitializePart1
                                                InitializePart2 MAKESYSTEM NextAffirmVersion
                                                ReInitializeAffirm SYSTEM\CHECK WRITE\DATE))
(DEFINEQ
```

97

```
(AffirmInitAfterMakesys
  [LAMBDA (FromSysout)                                    (* R.Bates " 5-Aug-80 18:56")

        (* * This routine is called exactly once each time AFFIRM starts up. One-time initialization actions should occur
        here.)


    (if ~FromSysout
        then                                              (* assume that this has been done before by the
                                                          initialize routine for sysout.)
            (for (x (!VALUE ← <AFFIRMMAKESYSFILE>)) in AFTERSYSOUTFORMS do (EVAL x)))
    (UNADVISE EVALQT)
    [ADVISE 'EVALQT 'BEFORE 'FIRST '(COND
             ((NOT (UserProfile (QUOTE BreakAccess)
                                T))
               (TAB 0 0 T)
               (PRIN1 "Please don't type Control-D; Returning to top-level" T)
               (TERPRI T)
               (ENTEREVALQT)
               (COND
                 (UsingTed (printout T .TAB0 0 "Stopping Affirm for TED." T)
                           (LOGOUT)))
               (AffirmExec))
             (T (printout T .TAB0 0 "To return to AFFIRM call AffirmExec." T]
    CHANGEDADVICELST←NIL
    CHANGEDVARSLST←NIL
    (ENTEREVALQT)                                         (* ENTEREVALQT is the first thing EVALQT calls, it
                                                          restores anything that was on RESETLST and evals the
                                                          things on RESETFORMS.)

    (AffirmExec])
```

98

```
(AffirmMakesys
  [LAMBDA NIL                                             (* R.Bates "19-Nov-80 10:15")
    (InitializePart2)
    (DRIBBLE)
    (ECHOCONTROL 27 'SIMULATE)
    (for x in BEFOREMAKESYSFORMS do (EVAL x) unless x:1='HERALD or (EDITFINDP x 'GREETFORM))
    (UNADVISE MAKESYS)
    (HERALD (CONCAT "AFFIRM-" (QUOTIENT AFFIRMVERSION 100)
                    "."
                    (IREMAINDER AFFIRMVERSION AFFIRMVERSION/100*100)
                    "  ("
                    (SUBSTRING (DATE)
                               1 2+(STRPOS '-(DATE)
                                           4))
                    ")"))
    (CLOSEALL)
    (ADVISE 'EVALQT 'BEFORE 'FIRST '(AffirmInitAfterMakesys))
    (MAKESYS AFFIRMMAKESYSFILE])
```

99

```
(AffirmSysoutInit
  [LAMBDA (FileName)                                      (* R.Erickson "15-Jan-81 15:55")

        (* * Called once per session, including when wake from MAKESYS (via AFTERSYSOUTFORMS))


    (PROG (notesFile)
          (if (FNTYP 'transcript)
              then (transcript NIL 'SYSOUT))
          (FIRSTNAME←NIL)
          (USERNAME←(MKATOM (USERNAME)))
          (SETNM 'AFFIRM)
          (GOODGUY←(SASSOC USERNAME INITIALSLST))
          (AFFIRMSYSOUTFILE←(COPY FileName):1)
```

```
        (CannedMessagesSeen←NIL)
        (InitializeUserProfile)
        (if AFFIRMINITFIRSTTIME
            then (LoadUserInitializationFile))
        (AFFIRMINITFIRSTTIME←NIL)
        (if ~(UserProfile 'BreakAccess T)
            then (for i in '(2 8) do (INTERRUPTCHAR i)
                                     (if i~=8
                                         then (ECHOCONTROL i 'UPARROW)))
          else (INTERRUPTCHAR T))
        (notesFile←(PACKFILENAME 'DIRECTORY PV\DIRECTORY 'EXTENSION AFFIRMVERSION 'BODY 'NOTE))
        (if (INFILEP notesFile)
            then (printout T .TAB0 0 notesFile " describes this version." T])
```

100

```
(InitializePart1
  [LAMBDA (Build)                                    (* R.Erickson "24-Sep-81 15:54")

          (* * Gets called once only, when system is made, after files are loaded.)


    (PUTPROP 'TTYCHARSET 'ExpressionWithType ':)
    (NoteInterfaces '(cases\Interface result\Interface))
    (SetHierarchy '(ExpressionWithType)
                  9)
    (XSet EQOP Equal)
    [if ~(MEMBER '(DWIMUSERFN)
                 DWIMUSERFORMS)
        then (push DWIMUSERFORMS '(DWIMUSERFN]
    ProfileReadTable←(COPYREADTABLE PASCAL\READ\TABLE)
    (SETBRK '(?)
            1 ProfileReadTable)
    (SETBRK '(27)
            0 ProfileReadTable)

          (* * Initialize cursor for proof tree)


    (MakeCurrent T)
    (if Build
        then (ReInitializeAffirm T)
             WHEREIS.HASH←(COPYHASHFILE WHEREIS.HASH (MKATOM (PACK* 'WHEREIS- AFFIRMVERSION '.HASH)))
             WHEREISHASHFILE←NIL])
```

101

```
(InitializePart2
  [LAMBDA NIL                                        (* R.Bates " 6-Feb-80 14:22")
    (LOADFNS '(IH\Induction Prop\Induction)
             (PACKFILENAME 'DIRECTORY PV\DIRECTORY 'NAME 'INDUCTION)
             T)
    (InitXevalDtvs '((1 512)
                     (8 5000)))
    (DeclareType 'Basis)
    (Edit 'Basis])
```

102

```
(MAKESYSTEM
  [LAMBDA (List Question1 Question2)                 (* R.Bates "19-Mar-80 11:07")

          (* * This routine loads system files during a system build.)


    (PROG (FILES COMFILES SystemLoad)
          (if List=NIL
              then List←PV\SYSTEM\FILES
                   SystemLoad←'SYSLOAD)
          (printout NIL .TAB0 0 "Checking the write dates..." T)
          (SYSTEM\CHECK List)
          (printout NIL T)
          (FILES←(DREVERSE FILES))
          (COMFILES←(DREVERSE COMFILES))
          [PROG (LISPXHIST ADDSPELLFLG BUILDMAPFLG (LOADDBFLG ('NO)))
                (if Question1='YES or Question1=NIL and (AFFIRMUSER NIL NIL
                                                        "Do you want to load AFFIRM main files? "
                                                                NIL T)='Y
```

```
                then (LOAD (PACKFILENAME 'DIRECTORY PV\DIRECTORY 'NAME 'INIT 'EXTENSION 'LISP)
                           SystemLoad)
                     (LOAD (PACKFILENAME 'DIRECTORY 'LISPUSERS 'NAME 'HASH 'EXTENSION 'COM)
                           SystemLoad)
                     (LOAD (PACKFILENAME 'DIRECTORY 'LISPUSERS 'NAME 'DATABASEFNS 'EXTENSION 'COM)
                           SystemLoad)
                     (LOAD (PACKFILENAME 'DIRECTORY 'LISPUSERS 'NAME 'WHEREIS 'EXTENSION 'COM)
                           SystemLoad)
                     (LOAD (PACKFILENAME 'DIRECTORY PV\DIRECTORY 'NAME 'INDEX 'EXTENSION 'COM)
                           SystemLoad)
                     (MOVD 'WHEREIS 'HashWHEREIS)
                     (MOVD 'AFFIRMWHEREIS 'WHEREIS))
           (if Question2='YES or Question2=NIL and (AFFIRMUSER NIL NIL
                                                   "Do you want to load the rest of the system? "
                                                          NIL T)='Y
                then (for I in COMFILES do (LOAD I SystemLoad]
      (NoteFiles FILES SystemLoad])
```

103

```
(NextAffirmVersion
  [LAMBDA NIL                                          (* R.Bates "11-Jul-80 14:12")
    (PROG (FONTCHANGEFLG File)
          (File←(OPENFILE VERSIONFILE 'INPUT))
          (AFFIRMVERSION←(READ File)+ 1)
          (AFFIRMMAKESYSFILE←(OUTFILEP (PACKFILENAME 'DIRECTORY PV\DIRECTORY 'NAME 'AFFIRM 'EXTENSION
                                                     (if SYSTEMTYPE='TOPS20
                                                         then 'EXE
                                                       else 'SAV)
                                                     'VERSION AFFIRMVERSION 'PROTECTION 777700)))
          (CLOSEF? File)
          (File←(OPENFILE VERSIONFILE 'OUTPUT))
          (printout File .I5 AFFIRMVERSION T)
          (CLOSEF? File)
          (RETURN AFFIRMVERSION])
```

104

```
(ReInitializeAffirm
  [LAMBDA (IncrCount)                                  (* R.Bates " 6-Feb-80 12:21")
    (if IncrCount
        then (NextAffirmVersion))
    (renumber LISPXHISTORY:4)
    AFFIRMINITFIRSTTIME←T
    (DefaultUserProfile)
    (forget)
    (RECLAIM])
```

105

```
(SYSTEM\CHECK
  [LAMBDA (LIST\OF\FILES)                              (* Edited by R.Bates on 19-DEC-77;
                                                       from version 108)
    (for (FILE TEMPF TEMPC) in LIST\OF\FILES do (TEMPF←TEMPC←NIL)
                                                (if TEMPF←(INFILEP (PACKFILENAME 'DIRECTORY FILE:1
                                                                                 'BODY FILE::1))=NIL
                                                    then (LISPXPRIN1 " Can not find source file ")
                                                         (LISPXPRIN1 FILE::1)
                                                         (LISPXTERPRI)
                                                  else FILES← <TEMPF ! FILES>)
                                                (if TEMPC←(INFILEP (PACKFILENAME
                                                                    'DIRECTORY FILE:1 'NAME
                                                                    (FILENAMEFIELD FILE::1 'NAME)
                                                                    'EXTENSION 'COM))=NIL
                                                    then (LISPXPRIN1 " Can not find com file ")
                                                         (LISPXPRIN1 (CONCAT FILE::1 ".COM"))
                                                         (LISPXTERPRI)
                                                  else COMFILES← <TEMPC ! COMFILES>)
                                                (if TEMPF and TEMPC
                                                    then (if (WRITE\DATE FILES:1) gt
                                                             (WRITE\DATE COMFILES:1)
                                                             then (LISPXPRIN1 FILES:1)
                                                                  (LISPXPRIN1
                                                                    " should be Bcompl again !!!")
                                                                  (LISPXTERPRI])
```

106

```
(WRITE\DATE
  [LAMBDA (FILE)

          (* * Given a File WRITE\DATE return the date the file was written as a number. To convert to a string call DATE.
          To convert a string which is a date call IDATE. The number is such that if date1 is before date2
          (both strings) then IDATE (date1) < IDATE (date2).)


    (PROG (TEMP JFN)
          (if (INFILEP FILE)
              then JFN←(GTJFN FILE NIL NIL 8590721024))
          (if JFN=NIL
              then (PRIN1 FILE)
                   (PRIN1 " not found")
                   (RETURN 0))
          (TEMP←(JSYS 51 JFN 262156 1 1))
          (RLJFN JFN)
          (RETURN TEMP])
)

(RPAQ AFFIRMSYSOUTFILE NIL)

(RPAQ CannedMessagesSeen NIL)

(RPAQ COMPILEHEADER " compiled for AFFIRM on ")

(RPAQ MKSWAPSIZE 0)

(RPAQ PRETTYHEADER " file created for AFFIRM on ")

(RPAQQ PV\BASIS\FILES ((AFFIRM . Boolean)
                       (AFFIRM . BUILTIN)
                       (AFFIRM . Integer)
                       (AFFIRM . TypeParameter)
                       (AFFIRM . Induction)
                       (AFFIRM . ProcedureCall)))

(RPAQ PV\DIRECTORY 'AFFIRM)

(RPAQQ PV\SYSTEM\FILES ((AFFIRM . AFFIRMEXEC)
                        (AFFIRM . CEVAL)
                        (AFFIRM . FORMULAIO)
                        (AFFIRM . HELP)
                        (AFFIRM . INFIXPRINT)
                        (AFFIRM . LOGIC)
                        (AFFIRM . PARSER)
                        (AFFIRM . PARSERHELPER)
                        (AFFIRM . PASCAL)
                        (AFFIRM . PROFILE)
                        (AFFIRM . REQUEST)
                        (AFFIRM . REWRITERULE)
                        (AFFIRM . SPECIFICATION)
                        (AFFIRM . SUFFICIENT)
                        (AFFIRM . TEDIT)
                        (AFFIRM . THEOREMPROVER)
                        (AFFIRM . TREE)
                        (AFFIRM . VCGEN)
                        (AFFIRM . XEVAL)))

(RPAQQ PVMAINTAINERS (DTHOMPSON))

(ADDTOVAR INITIALSLST (BAKER . D.Baker)
                      (DTHOMPSON . D.Thompson)
                      (ERICKSON . R.Erickson)
                      (GERHART . S.Gerhart)
                      (LONDON . R.London)
                      (MUSSER . D.Musser)
                      (RBATES . R.Bates)
                      (TAYLOR . D.Taylor)
                      (WILE . D.Wile))
[if ~ (EDITFINDP AFTERSYSOUTFORMS 'AffirmSysoutInit)
    then
    (/SET (QUOTE AFTERSYSOUTFORMS)
          (APPEND AFTERSYSOUTFORMS (LIST (QUOTE (INIT\BACKSPACE))
                                         (QUOTE (AffirmSysoutInit !VALUE]
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 10b Name definitions -- predefined, hidden, reserved, known. Depends on variables in 10a)]

```
(RPAQQ PredefinedNames ((AFFIRMObjects (affirmobjects . AFFIRMObjects)
                                       (affirmobject . AFFIRMObjects)
                                       (object . AFFIRMObjects)
                                       Arcs
                                       (arcs . Arcs)
                                       (arc . Arcs)
                                       Axioms
                                       (axioms . Axioms)
                                       (axiom . Axioms)
                                       Commands
                                       (commands . Commands)
                                       (command . Commands)
                                       Definitions
                                       (definitions . Definitions)
                                       (definition . Definitions)
                                       (defn. . Definitions)
                                       Directories
                                       (directories . Directories)
                                       (directory . Directories)
                                       DiscardOptions
                                       (discardoption . DiscardOptions)
                                       (discardoptions . DiscardOptions)
                                       Disconnected
                                       (disconnected . Disconnected)
                                       Files
                                       (files . Files)
                                       (file . Files)
                                       FileTypes
                                       (filetypes . FileTypes)
                                       (filetype . FileTypes)
                                       Groups
                                       (groups . Groups)
                                       (group . Groups)
                                       HelpTopics
                                       (helptopics . HelpTopics)
                                       (helptopic . HelpTopics)
                                       History Interfaces (interfaces . Interfaces)
                                       (interface . Interfaces)
                                       Lemmas
                                       (lemmas . Lemmas)
                                       (lemma . Lemmas)
                                       (rulelemmas . Lemmas)
                                       (rulelemma . Lemmas)
                                       Lhs
                                       (lhs . Lhs)
                                       Nodes
                                       (nodes . Nodes)
                                       (node . Nodes)
                                       (Theorems . Nodes)
                                       (Theorem . Nodes)
                                       (theorems . Nodes)
                                       (theorem . Nodes)
                                       PrintObjects
                                       (printobjects . PrintObjects)
                                       (printobject . PrintObjects)
                                       ProfileEntries
                                       (profileentries . ProfileEntries)
                                       (profileentry . ProfileEntries)
                                       Schemas
                                       (schemas . Schemas)
                                       (schema . Schemas)
                                       Types
                                       (types . Types)
                                       (type . Types)
                                       TypeParts
                                       (typeparts . TypeParts)
                                       (typepart . TypeParts)
                                       Variables
                                       (variables . Variables)
                                       (variable . Variables)
                                       Unexpected
                                       (unexpected . Unexpected))
        NIL NIL
        (; @ abort adopt affirmed? annotate apply arc assume augment automatic axiom cases choose
           clear compile complete declare define denote discard distinct down e edit employ end enter
           eval exec fix forget freeze genvcs gripe help infix interface invoke keep let lisp load
           name needs next nochange normalize normint note ok print profile put quit read readp redo
           renumber replace resume retry review rulelemma save schema search set split stop storage
```

```
      sufficient? suppose swap thaw theorem transcript try type undo up use (axioms . axiom)
      (interfaces . interface)
      (rulelemmas . rulelemma)
      (schemas . schema)
      (automatics . automatic))
   NIL
   ((DIRECTORYNAME)
    (DIRECTORYNAME T)
    PV\Library PV\DIRECTORY)
   NIL
   (Source DatedSource Saved Compiled OldCompiled OldSaved)
   NIL NIL NIL NIL NIL (? (assumption . assumptions)
                          assumptions BadEquations (badequations . BadEquations)
                          both file history IH (ih . IH)
                          implist known lhs named names next (node . prop)
                          original parts proof (proofs . proof)
                          prop
                          (propn . prop)
                          (propns . prop)
                          (props . prop)
                          result status type (used . uses)
                          uses
                          (variable . variables)
                          variables)
   NIL NIL (Basis any Boolean BUILTIN Integer TypeParameter Induction ProcedureCall)
   (needs (need . needs)
          declare
          (declaration . declare)
          (declarations . declare)
          interface
          (interfaces . interface)
          macro
          (macros . macro)
          axiom
          (axioms . axiom)
          lemma
          (lemmas . lemma)
          defn.
          (define . defn.)
          (definition . defn.)
          (definitions . defn.)
          automatic
          (auto . automatic)
          (automatics . automatic)
          schema
          (schemas . schema)
          distinct nochange)
   NIL NIL NIL (? Disconnected History Interfaces Lhs Theorems Variables)))

(RPAQ HiddenNames (create KnownNames AFFIRMObjects ← (QUOTE (Unexpected))
                          Commands ←
                          (QUOTE (; batch copy enter ImplicitE keep lemma macro monitor program rule
                                    show types))
                          FileTypes ← (QUOTE (OldCompiled OldSaved))
                          Types ← (QUOTE (BUILTIN Induction TypeParameter))))

(RPAQ ReservedNames (create KnownNames Types ← (QUOTE (Interface Schema))))

(RPAQ KnownNames (create KnownNames PrintObjects ← (APPEND (fetch PrintObjects of PredefinedNames)
                                                           (COPY (fetch TypeParts of PredefinedNames))
                                                           )
                         USING PredefinedNames))
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 11 Advice) ]

(PUTPROPS PRINT ARGNAMES (X FILE RDTBL))

(PUTPROPS PRINT-IN-COMPILE1A READVICE [(COMPILE1A . PRINT)
                                       (AROUND NIL (COND ((AND (FGETD (QUOTE SuppressAdvisedPrint?))
                                                               (SuppressAdvisedPrint? (QUOTE
                                                                                        COMPILE1A)))
                                                          X)
                                                         (T *])

(PUTPROPS PRINT-IN-LAPBLOCK READVICE [(LAPBLOCK . PRINT)
                                      (AROUND NIL (COND ((AND (FGETD (QUOTE SuppressAdvisedPrint?))
                                                              (SuppressAdvisedPrint? (QUOTE LAP)))
                                                         X)
                                                        (T *])
(READVISE PRINT-IN-COMPILE1A PRINT-IN-LAPBLOCK)

(PUTPROPS LOGOUT ARGNAMES NIL)

(PUTPROPS LOGOUT READVICE (NIL (AROUND NIL (EvalFormWithOutDribble *))))

(PUTPROPS SAVESET READVICE [NIL (BEFORE NIL (COND ((FMEMB NAME NOSAVESETVARS)
                                                   (SETQ FLG (QUOTE NOPROPSAVE])

(PUTPROPS SUBSYS READVICE (NIL (AROUND NIL (EvalFormWithOutDribble *))))
(READVISE LOGOUT SAVESET SUBSYS)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 12 XIVUS) ]

(RPAQQ NEGINF NEGINF)

(RPAQQ POSINF POSINF)

(RPAQQ NEWNAMESVARS (ALLIO ANDIO DIFFIO DIVIO EQVIO EQIO GEIO GTIO IMPIO INVIO LEIO LTIO MAXIO MINIO
                           NEGIO MODIO NEIO NEWNAMES NOTIO ORIO ADDIO QUOTIENTIO SOMEIO MULTIO))

(RPAQQ ALLIO ALL)

(RPAQQ ANDIO AND)

(RPAQQ DIFFIO DIFFERENCE)

(RPAQQ DIVIO DIV)

(RPAQQ EQVIO EQV)

(RPAQQ EQIO Equal)

(RPAQQ GEIO GE)

(RPAQQ GTIO GT)

(RPAQQ IMPIO IMP)

(RPAQQ INVIO INVERSE)

(RPAQQ LEIO LE)

(RPAQQ LTIO LT)

(RPAQQ MAXIO MAX)

(RPAQQ MINIO MIN)

(RPAQQ NEGIO MINUS)

(RPAQQ MODIO MOD)

(RPAQQ NEIO NE)

(RPAQQ NEWNAMES ((MAXOP MAXIO)
                 (MINOP MINIO)
                 (IMPOP IMPIO)
                 (EQVOP EQVIO)
                 (NOTOP NOTIO)
                 (NEOP NEIO)
                 (LEOP LEIO)
                 (GEOP GEIO)
                 (OROP ORIO)
                 (ANDOP ANDIO)
                 (ADDOP ADDIO)
                 (DIFFOP DIFFIO)
                 (NEGOP NEGIO)
                 (MULTOP MULTIO)
                 (INVOP INVIO)
                 (QUOTIENTOP QUOTIENTIO)
                 (LTOP LTIO)
                 (GTOP GTIO)
                 (EQOP EQIO)
                 (DIVOP DIVIO)
                 (MODOP MODIO)
                 (ALLOP ALLIO)
                 (SOMEOP SOMEIO)))

(RPAQQ NOTIO NOT)

(RPAQQ ORIO OR)

(RPAQQ ADDIO PLUS)

(RPAQQ QUOTIENTIO QUOTIENT)

(RPAQQ SOMEIO SOME)

(RPAQQ MULTIO TIMES)
[DECLARE: EVAL@COMPILE
```

```
(TYPERECORD ArrayAccess (Array Index))

(TYPERECORD ArrayWrite (Array Index Value))

(TYPERECORD Assign (baseName qualifiers newValue))

(TYPERECORD HeapOrFileAccess (HeapOrFile Pointer)
                             HeapOrFile ←(QUOTE HeapOrFileSpace))

(TYPERECORD HeapOrFileWrite (HeapOrFile Pointer Value))

(TYPERECORD RecordAccess (Record Field))

(TYPERECORD RecordWrite (Record Field Value))

(ATOMRECORD priorityProp (priority))

(RECORD priorityRecord (left . right))
]

(RPAQQ UTILITYRECORDS (XEVAL\ARG XEVAL\EL XEVAL\MEM))
[DECLARE: EVAL@COMPILE

(RECORD XEVAL\ARG (OPR . ARGS)
                  (RECORD ARGS (ARG1 ARG2 ARG3)))

(RECORD XEVAL\EL (EL1 EL2 EL3))

(RECORD XEVAL\MEM (MEM1 . MEM2))
]
(#INIT\XEVAL)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* 13 compiled files and CLISP)]

```
(LOADCOMP (QUOTE <affirm>parser))
(CLISPDEC (QUOTE FAST))

(RPAQ UPDATEMAPFLG NIL)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 14 LISPX) ]

(ADDTOVAR LISPXMACROS (exec (PROGN (OR (NLSETQ (SETQ LASTEXEC (SUBSYS LASTEXEC)))
                                       (SETQ LASTEXEC (SUBSYS)))
                                   LASTEXEC))
                      (qu (LOGOUT))
                      (quit (LOGOUT))
                      (TCOPY (for x in LISPXLINE collect (TypeFile x)))
                      (XED (SetupXed)))

(ADDTOVAR LISPXCOMS exec qu quit TCOPY XED)

(ADDTOVAR LISPXMACROS (EXEC (PROGN (OR [PROG (HELPDEPTH)
                                             (RETURN (NLSETQ (SETQ LASTEXEC (SUBSYS LASTEXEC]
                                       (SETQ LASTEXEC (SUBSYS)))
                                   LASTEXEC))
                      (TECO (PROGN [COND ([NOT (PROG (HELPDEPTH)
                                                     (RETURN (NLSETQ (SETQ LASTTECO (SUBSYS LASTTECO]
                                           (SETQ LASTTECO (SUBSYS (QUOTE HTECO]
                                   LASTTECO))
                      (ok (RETFROM (OR (STKPOS (QUOTE USEREXEC))
                                       (QUOTE LISPX))
                                   T T)))

(ADDTOVAR LISPXCOMS TECO)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

```
(* 15 Macros, info) ]

(DECLARE: EVAL@COMPILE

(PUTPROPS CONSIT MACRO [X (LIST (QUOTE SETQ)
                                (CADR X)
                                (LIST (QUOTE CONS)
                                      (CAR X)
                                      (CADR X])

(PUTPROPS Dprintout MACRO (X (DprintoutMacro X)))

(PUTPROPS EQCAR MACRO [LAMBDA (U V)
                      (AND (NOT (ATOM U))
                           (EQ (CAR U)
                               V])

(PUTPROPS EvalFormWithOutDribble MACRO (X (EvalFormWithOutDribbleMacro X)))

(PUTPROPS IfThenElse MACRO NIL)

(PUTPROPS NoSpreadApply* MACRO [X
          (CONS (QUOTE SELECTQ)
                (CONS (CADR X)
                      (for i from 0 to (COND
                                         ((CADDR X))
                                         (T 8))
                         collect [LIST i (CONS (QUOTE APPLY*)
                                               (CONS (CAR X)
                                                     (for j from 2 to i collect (LIST (QUOTE ARG)
                                                                                      (CADR X)
                                                                                      j]
                         finally (SETQ $$VAL (APPEND $$VAL (LIST (LIST (QUOTE APPLY)
                                                                       (CAR X)
                                                                       (LIST (QUOTE for)
                                                                             (QUOTE i)
                                                                             (QUOTE from)
                                                                             2
                                                                             (QUOTE to)
                                                                             (CADR X)
                                                                             (QUOTE collect)
                                                                             (LIST (QUOTE ARG)
                                                                                   (CADR X)
                                                                                   (QUOTE i])

(PUTPROPS PRINTHELP MACRO [LAMBDA (X)
                          (COND
                            (X (PRIN1 X])

(PUTPROPS PRINTLINES MACRO [X (LIST (QUOTE PROGN)
                                    [COND
                                      ((STRINGP (CAR X))
                                        (LIST (QUOTE PRIN1)
                                              (CAR X)))
                                      ((EQ T (CAR X))
                                        (QUOTE (TERPRI)))
                                      (T (LIST (QUOTE PRINTHELP)
                                               (CAR X]
                                    [COND
                                      ((AND (CDR X)
                                            (NEQ (CAR X)
                                                 T))
                                        (QUOTE (PRIN1 " "]
                                    (AND (CDR X)
                                         (CONS (QUOTE PRINTLINES)
                                               (CDR X])

(PUTPROPS TranslateApply MACRO (X (TranslateApplyMacro X)))

(PUTPROPS UCI\SUBST MACRO [LAMBDA (X Y Z)
                          (COND
                            ((EQUAL Y Z)
                              X)
                            (T (SUBST X Y Z])

(PUTPROPS UserProfile MACRO (x (UserProfileMacro x)))
)
(SETTEMPLATE (QUOTE ADDTOLIST)
             (QUOTE ((IF (EQ (CAR EXPR)
                             (QUOTE QUOTE))
                         (NIL SET)
```

```
                        EVAL)
                    EVAL PPE)))
(SETTEMPLATE (QUOTE DROPFROMLIST)
             (QUOTE ((IF (EQ (CAR EXPR)
                             (QUOTE QUOTE))
                         (NIL SET)
                         EVAL)
                    EVAL PPE)))
(SETTEMPLATE (QUOTE Dprintout)
             (QUOTE MACRO))
(SETTEMPLATE (QUOTE MapExpr)
             (QUOTE (EXPR FUNCTION . PPE)))
(SETTEMPLATE (QUOTE Minimize)
             (QUOTE (EVAL FUNCTION . PPE)))
(SETTEMPLATE (QUOTE NoSpreadApply*)
             (QUOTE MACRO))
(SETTEMPLATE (QUOTE TranslateApply)
             (QUOTE MACRO))
(SETTEMPLATE (QUOTE TranslateApply)
             (QUOTE MACRO))

(PUTPROPS IfThenElse INFO EVAL)

(PUTPROPS PRINTLINES INFO EVAL)

(PUTPROPS EvalFormWithOutDribble INFO EVAL)

(PUTPROPS MapExpr INFO EVAL)
(DECLARE: DOEVAL@COMPILE DONTCOPY

(ADDTOVAR GLOBALVARS ADDIO ALLIO ANDIO Ampersand Asterisk AtSign Blank CR Colon Comma DIFFIO DIVIO
          DoubleQuote EQIO EQVIO Ellipses EqualSign Escape ExclamationPoint FALSE GEIO GLOBALVAR GTIO
          Hyphen Hyphens IEXT IMPIO INVIO KnownNames LEIO LF LTIO LeftCurlyBracket LeftParenthesis
          MAXIO MINIO MODIO MULTIO NEGIO NEIO NOTIO NullString ORIO PARSERTRACE PercentSign Period
          Period QOP QUOTIENTIO QuestionMark RESETVARSLST RightCurlyBracket RightParenthesis SOMEIO
          SemiColon SingleQuote Slash SquareBracket Stars TRUE TabChar Terminal TypeSeparator
          (RuleTypes RuleList))
)
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA EvalFormWithOutDribble PRINTLINES)

(ADDTOVAR NLAML XSet BEEHIVE Report)

(ADDTOVAR LAMA )
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (17924 19484 (EndinColon 17936 . 18171) (EndsColon 18175 . 18308) (Skim 18312 . 18541) (
UCase\ToList 18545 . 19481)) (19643 23111 (ExtendName 19655 . 19878) (ExtendUsingList 19882 . 20892) (
Extension 20896 . 21752) (MakeExtension 21756 . 21925) (Shorten 21929 . 22658) (ShortenLessPrimes
22662 . 23108)) (23259 35361 (CallSubsys 23271 . 25911) (FtpFile 25915 . 28706) (GtjfnHelp 28710 .
32002) (InitializeLongGtjfn 32006 . 33423) (NotifyMailer 33427 . 34066) (SetupXed 34070 . 34277) (
TTYINFILE 34281 . 34441) (TTYOUTFILE 34445 . 34607) (setBlockE 34611 . 34679) (stringPack 34683 .
35358)) (35493 40897 (DprintoutMacro 35505 . 35859) (EvalFormWithOutDribbleMacro 35863 . 36043) (
Report 36047 . 39080) (TranslateApplyMacro 39084 . 39911) (UserProfileMacro 39915 . 40894)) (40938
43092 (InitXevalDtvs 40950 . 43089)) (43327 50086 (/DREMASSOC 43339 . 44398) (ADDTOLIST 44402 . 44719)
 (AffirmSORT 44723 . 45051) (Closure 45055 . 45598) (Combinations 45602 . 46205) (DROPFROMLIST 46209 .
 46498) (ElementsIn 46502 . 46787) (FoundIn 46791 . 46997) (Minimize 47001 . 47548) (
OccursAsOperatorIn 47552 . 48003) (POS 48007 . 48348) (REMOVEDUPLICATES 48352 . 48419) (Separate 48423
 . 48986) (SubstWhenEq 48990 . 49171) (UCI\SUBST 49175 . 49353) (UNMKLIST 49357 . 49630) (UPTO 49634 .
 49878) (firstElement 49882 . 50083)) (50219 53542 (CloseAndOpenDribble 50231 . 50723) (AFFIRMMAPRINT
50727 . 51864) (BEEHIVE 51868 . 52127) (EnhanceHp 52131 . 52582) (Heading 52586 . 52657) (PRINTLINES
52661 . 53247) (EvalFormWithOutDribble 53251 . 53539)) (53666 58635 (#INIT\XEVAL 53678 . 57025) (
BINXEVAL 57029 . 57183) (DEFINE\NARY\OP 57187 . 57394) (DEFINE\OPR 57398 . 57830) (N2BINARY 57834 .
58216) (N2BINARY1 58220 . 58468) (SIMPLIFY\BREAK 58472 . 58522) (XSet 58526 . 58632)) (59139 86742 (
AFFIRMUSER 59151 . 61214) (AFFIRMWHEREIS 61218 . 62376) (AddToFile 62380 . 62699) (Affirm/ADDPROP
62703 . 63059) (BadFile 63063 . 63551) (CHRCT 63555 . 63612) (CallEDITF 63616 . 64598) (CallEditor
64602 . 67972) (CheckLoad 67976 . 70077) (CloseDribble 70081 . 70522) (CorrectOneFileDate 70526 .
71001) (DWIMUSERFN 71005 . 72177) (Debug 72181 . 72780) (EQCAR 72784 . 72842) (EXTENT 72846 . 73051) (
FillOut 73055 . 73657) (GenIndex 73661 . 75444) (IsEq 75448 . 75737) (LexOrder 75741 . 76498) (MapExpr
 76502 . 76909) (MatchArg 76913 . 77382) (MatchOp 77386 . 77840) (PagesLeft 77844 . 78259) (
PagesWarning 78263 . 79283) (RecoverTranscript 79287 . 80522) (TypeFile 80526 . 81499) (UpdateAffirm
81503 . 82598) (baseNameSubst 82602 . 83479) (exec 83483 . 84079) (forget 84083 . 84368) (ruleSeqParse
 84372 . 85472) (storage 85476 . 86739)) (88203 93937 (AffirmBacktrace 88215 . 89457) (AffirmBreak
89461 . 90813) (AffirmError 90817 . 91950) (ParsingError 91954 . 92500) (Unexpected 92504 . 93934)) (
94998 103735 (AffirmInitAfterMakesys 95010 . 96282) (AffirmMakesys 96286 . 96947) (AffirmSysoutInit
96951 . 98112) (InitializePart1 98116 . 99034) (InitializePart2 99038 . 99369) (MAKESYSTEM 99373 .
101014) (NextAffirmVersion 101018 . 101695) (ReInitializeAffirm 101699 . 101994) (SYSTEM\CHECK 101998
. 103091) (WRITE\DATE 103095 . 103732)))))
STOP
```