```
-- File DebugUtilities.mesa
-- Last edited by
--                Sandman; April 24, 1978  4:43 PM
--                Barbara; June 28, 1978  9:40 AM

DIRECTORY
  CommandDefs: FROM "commanddefs" USING [WriteSignalString],
  ControlDefs: FROM "controldefs" USING [FrameHandle, GlobalFrameHandle],
  CoreSwapDefs: FROM "coreswapdefs" USING [SVPointer],
  DebugBreakptDefs: FROM "debugbreakptdefs" USING [
    SourceFileMissing, StringMatchFailure, StringTooLong],
  DebugContextDefs: FROM "debugcontextdefs" USING [
    IncorrectVersion, InvalidGlobalFrame, InvalidImageFile, InvalidPSB,
    WriteGlobalContext, WriteLocalContext],
  DebugData: FROM "debugdata" USING [
    gContext, lContext, level, pContext, StatePtr],
  DebuggerDefs: FROM "debuggerdefs" USING [
    AmIaRecord, ClobberedAccessLink, ClobberedFrame, FormatRecord,
    NoPreviousFrame, SymbolTableNotFound],
  DebugMiscDefs: FROM "debugmiscdefs" USING [commander],
  DebugSymbolDefs: FROM "debugsymboldefs" USING [CheckDCache],
  DebugUtilityDefs: FROM "debugutilitydefs" USING [
    CoreSwap, DebuggerError, InvalidAddress, LoadStateInvalid, MREAD,
    NonExistentMemoryPage, NoUserProcsLoaded],
  ImageDefs: FROM "imagedefs" USING [StopMesa],
  IODefs: FROM "iodefs" USING [
    CR, LineOverflow, NewLine, NUL, Rubout, WriteChar, WriteOctal,
    WriteString],
  Mopcodes: FROM "mopcodes",
  ProcessDefs: FROM "processdefs" USING [Aborted, CurrentPSB],
  StringDefs: FROM "stringdefs" USING [InvalidNumber],
  SymbolTableDefs: FROM "symboltabledefs" USING [NoSymbolTable],
  SystemDefs: FROM "systemdefs" USING [
    AllocateHeapNode, AllocateHeapString, FreeHeapNode, FreeHeapString],
  TrapDefs: FROM "trapdefs" USING [ResumeError];

DebugUtilities: PROGRAM
IMPORTS DDptr: DebugData, CommandDefs, DebugBreakptDefs, DebugContextDefs,
  DebuggerDefs, DebugMiscDefs, DebugSymbolDefs, DebugUtilityDefs, IODefs,
  ProcessDefs, StringDefs, SymbolTableDefs, SystemDefs, TrapDefs
EXPORTS DebugUtilityDefs, DebugMiscDefs
SHARES ProcessDefs =

BEGIN

DebugProceed: PUBLIC SIGNAL = CODE;
DebugAbort: PUBLIC SIGNAL = CODE;
ControlDEL: PUBLIC SIGNAL = CODE;
Quit: PUBLIC SIGNAL = CODE;
LookupFail: PUBLIC SIGNAL [s: STRING] = CODE;
KillSession: PUBLIC SIGNAL = CODE;
CommandNotAllowed: PUBLIC SIGNAL = CODE;

resetdebugger: PROCEDURE [sc: ControlDefs.FrameHandle,
  sg: ControlDefs.GlobalFrameHandle, ss: CoreSwapDefs.SVPointer] =
  BEGIN
  DDptr.level ← DDptr.level -1;
  DDptr.StatePtr ← ss;
  IF sc # NIL THEN DebugContextDefs.WriteLocalContext[sc]
  ELSE IF sg # NIL THEN DebugContextDefs.WriteGlobalContext[sg]
  ELSE DDptr.pContext ← DDptr.lContext ← DDptr.gContext ← NIL;
  RETURN
  END;

DebugCommand: PUBLIC PROCEDURE [sp: CoreSwapDefs.SVPointer] =
  BEGIN
  OPEN DebugBreakptDefs, DebugMiscDefs, IODefs, DebugUtilityDefs, DebuggerDefs, DebugContextDefs, Comma
**ndDefs;
  savlcontext: ControlDefs.FrameHandle ← DDptr.lContext;
  savgcontext: ControlDefs.GlobalFrameHandle ← DDptr.gContext;
  savStatePtr: CoreSwapDefs.SVPointer ← DDptr.StatePtr;
  FR: FormatRecord ←
    FormatRecord[indentation: 2, symid: TRUE, firstsym: TRUE,
      symdelim: '=, intersym: CR, startchar: NUL, termchar: NUL];

  DDptr.level ← DDptr.level + 1;
  DDptr.StatePtr ← sp;
  WriteLocalContext[MREAD[@sp.dest]];
  DDptr.pContext ← MREAD[ProcessDefs.CurrentPSB];
  DO
    BEGIN
    CheckDStrings[]; DebugSymbolDefs.CheckDCache[];
    WriteEOL[]; FR.firstsym ← TRUE;
    THROUGH [1..DDptr.level] DO WriteChar['>] ENDLOOP;
    commander[LOOPHOLE[MREAD[@sp.dest]], MREAD[ProcessDefs.CurrentPSB] !
      Rubout =>
        BEGIN WriteSignalString[del]; CONTINUE END;
      CommandNotAllowed =>
        BEGIN WriteSignalString[notallowed]; CONTINUE END;
      ControlDEL =>
        BEGIN WriteSignalString[aborted]; CONTINUE END;
      DebugAbort, ProcessDefs.Aborted => CONTINUE;
      SymbolTableNotFound, SymbolTableDefs.NoSymbolTable =>
        BEGIN WriteEOL[]; WriteSignalString[nosymtab]; CONTINUE END;
      LookupFail =>
        BEGIN WriteEOL[]; WriteChar['!]; WriteString[s]; CONTINUE END;
      StringTooLong, LineOverflow =>
        BEGIN WriteEOL[]; WriteSignalString[toolong]; CONTINUE END;
      SourceFileMissing =>
        BEGIN
        WriteEOL[];
        WriteSignalString[file];
        IF sourcename # NIL THEN WriteString[sourcename]
        ELSE WriteSignalString[compress];
        CONTINUE
        END;
      StringMatchFailure =>
        BEGIN WriteEOL[]; WriteSignalString[string]; WriteString[s]; CONTINUE END;
      StringDefs.InvalidNumber =>
        BEGIN WriteEOL[]; WriteSignalString[num]; CONTINUE END;
      KillSession => GOTO kill;
      Quit => GOTO abort;
      DebugProceed => EXIT;
      UNWIND => resetdebugger[savlcontext, savgcontext, savStatePtr];
      LoadStateInvalid =>
        BEGIN WriteEOL[]; WriteSignalString[notallowed]; CONTINUE END;
      InvalidAddress =>
        BEGIN WriteEOL[]; WriteSignalString[notallowed]; CONTINUE END;
      NonExistentMemoryPage =>
        BEGIN WriteEOL[]; WriteSignalString[notallowed]; CONTINUE END;
      TrapDefs.ResumeError =>
        BEGIN WriteSignalString[resume]; CONTINUE END;
      ClobberedFrame =>
        BEGIN WriteOctal[f]; WriteSignalString[clobfr]; CONTINUE END;
      NoPreviousFrame =>
        BEGIN WriteOctal[f]; WriteSignalString[null]; CONTINUE END;
      InvalidGlobalFrame =>
        BEGIN WriteOctal[f]; WriteSignalString[ngf]; CONTINUE END;
      InvalidImageFile =>
        BEGIN WriteString[image]; WriteSignalString[nimage]; CONTINUE END;
      IncorrectVersion =>
        BEGIN WriteString[file]; WriteSignalString[version]; RESUME END;
      InvalidPSB =>
        BEGIN WriteOctal[psb]; WriteSignalString[npsb]; CONTINUE END;
      ClobberedAccessLink =>
        BEGIN WriteOctal[f]; WriteSignalString[cloba]; CONTINUE END;
      AmIaRecord => RESUME[FALSE];
      NoUserProcsLoaded =>
        BEGIN WriteSignalString[notloaded]; CONTINUE END
      ];
    EXITS
      kill =>
        BEGIN
        resetdebugger[savlcontext, savgcontext, savStatePtr];
        CoreSwap[kill !
          DebuggerError => ImageDefs.StopMesa[];
          DebugAbort => LOOP];
        SIGNAL DebugAbort;
        END;
      abort =>
        BEGIN
        resetdebugger[savlcontext, savgcontext, savStatePtr];
        CoreSwap[quit ! DebuggerError => ImageDefs.StopMesa[]];
        SIGNAL DebugAbort;
        END;
      dontquit, continue => NULL;
    END;
    ENDLOOP;
  resetdebugger[savlcontext, savgcontext, savStatePtr];
  RETURN
  END;

WriteCharZ: PUBLIC PROCEDURE [c: CHARACTER] =
  BEGIN OPEN IODefs; IF c=NUL THEN RETURN; WriteChar[c]; RETURN END;

WriteEOL: PUBLIC PROCEDURE =
  BEGIN OPEN IODefs; IF ~NewLine[] THEN WriteChar[CR]; RETURN END;

DStringItem: TYPE = RECORD [
  next: POINTER TO DStringItem,
  string: STRING,
  level: INTEGER];

DStringList: POINTER TO DStringItem ← NIL;

DGetString: PUBLIC PROCEDURE [n: INTEGER] RETURNS [s: STRING] =
  BEGIN OPEN SystemDefs;
  dl: POINTER TO DStringItem;
  s ← AllocateHeapString[n];
  dl ← AllocateHeapNode[SIZE[DStringItem]];
  dl↑ ← DStringItem[next: DStringList, level: DDptr.level, string: s];
  DStringList ← dl;
  RETURN
  END;

DFreeString: PUBLIC PROCEDURE [s: STRING] =
  BEGIN OPEN SystemDefs;
  pdl: POINTER TO DStringItem ← NIL;
  dl: POINTER TO DStringItem ← DStringList;
  UNTIL dl = NIL DO
    IF dl.string = s THEN
      BEGIN
      IF pdl = NIL THEN DStringList ← dl.next ELSE pdl.next ← dl.next;
      SystemDefs.FreeHeapString[s];
      FreeHeapNode[dl];
      RETURN
      END;
    pdl ← dl; dl ← dl.next;
    ENDLOOP;
  RETURN
  END;

CheckDStrings: PROCEDURE =
  BEGIN
  next, dl: POINTER TO DStringItem;
  pdl: POINTER TO DStringItem ← NIL;
  FOR dl ← DStringList, next UNTIL dl = NIL DO
    IF dl.level >= DDptr.level THEN
      BEGIN
      IF pdl = NIL THEN DStringList ← dl.next ELSE pdl.next ← dl.next;
      SystemDefs.FreeHeapString[dl.string];
      next ← dl.next;
      SystemDefs.FreeHeapNode[dl];
      END
    ELSE BEGIN next ← dl.next; pdl ← dl; END;
    ENDLOOP;
  RETURN
  END;

DebugUtilitiesInit: PUBLIC PROCEDURE =
  BEGIN
  DDptr.level ← 0;
  DebugSymbolDefs.CheckDCache[];
  CheckDStrings[];
  RETURN
  END;

END...
```