```
-- File DebugInit.mesa
-- Edited by:
--          Johnsson, August 15, 1978  9:37 AM
--          Sandman, April 17, 1978  9:04 AM
--          Barbara, June 20, 1978  1:32 PM

DIRECTORY
  AltoDefs: FROM "altodefs" USING [BYTE],
  ControlDefs: FROM "controldefs" USING [
    BytePC, FrameHandle, GFT, GFTIndex, GlobalFrameHandle, NullFrame,
    ProcDesc, SD, SignalDesc, StateVector],
  CoreSwapDefs: FROM "coreswapdefs" USING [SVPointer],
  DebugBreakptDefs: FROM "debugbreakptdefs" USING [
    BBPointer, BTtype, ClearBreakBlocks, EXOItype, FindBPRec, FindSegBPRec],
  DebugCacheDefs: FROM "debugcachedefs" USING [ReinitMap],
  DebugData: FROM "debugdata" USING [
    caseignoring, debugPilot, ESV, gContext, lContext, pContext,
    restartmessage, sigGF, StatePtr, userwindow, worrybreaks],
  DebuggerDefs: FROM "debuggerdefs" USING [
    AmIaRecord, BodySei, DumpLocals, DumpParameters, DumpRetVals, DumpSource,
    DumpValsFromState, FormatRecord, FrameRelBPC, ListOptions, SeiHandle,
    SigSeiHandle, SymFrameHandle, WriteFrameLocus, WriteTransferName],
  DebugMiscDefs: FROM "debugmiscdefs" USING [DebugCommand, WriteEOL],
  DebugSymbolDefs: FROM "debugsymboldefs" USING [
    DAcquireSymbolTable, DCheckSymbolItems, DReleaseSymbolTable,
    PurgeUserSymbols, SymbolsForGFrame],
  DebugUtilityDefs: FROM "debugutilitydefs" USING [
    AREAD, CheckFrame, CoreSwap, DebugUtilitiesInit, InitDebugger,
    InitializeDrum, InitMapCleanup, InitUCSHandler, InvalidateFileCache,
    MREAD, UserWriteString, MWRITE],
  IODefs: FROM "iodefs" USING [
    CR, NUL, ReadChar, WriteChar, WriteLine, WriteOctal, WriteString],
  OsStaticDefs: FROM "osstaticdefs" USING [OsStaticRecord, OsStatics],
  SDDefs: FROM "sddefs" USING [sSignal],
  StreamDefs: FROM "streamdefs" USING [StreamIndex],
  SymDefs: FROM "symdefs" USING [ISEIndex],
  TimeDefs: FROM "timedefs" USING [AppendDayTime, DefaultTime, UnpackDT],
  WindowDefs: FROM "windowdefs" USING [
    GetCurrentDisplayWindow, PaintDisplayWindow, WindowHandle];

DebugInit: PROGRAM [herald: STRING]
IMPORTS DebugCacheDefs, DDptr: DebugData, DebugMiscDefs,
  DebugBreakptDefs, DebugSymbolDefs, DebugUtilityDefs,
  DebuggerDefs, IODefs, TimeDefs, WindowDefs
EXPORTS DebugMiscDefs, DebugBreakptDefs =

BEGIN

SaveSignallerGF: PUBLIC PROCEDURE =
  BEGIN OPEN ControlDefs, DebugUtilityDefs;
  gfi: GFTIndex;
  gfi ← LOOPHOLE[MREAD[@SD[SDDefs.sSignal]], ProcDesc].gfi;
  DDptr.sigGF ← MREAD[@GFT[gfi].frame];
  RETURN;
  END;

BreakInstToState: PUBLIC PROCEDURE [sp: CoreSwapDefs.SVPointer, b: AltoDefs.BYTE] =
  BEGIN OPEN ControlDefs;
  state: StateVector;
  Dstate: DESCRIPTOR FOR ARRAY OF UNSPECIFIED ← DESCRIPTOR[@state, SIZE[StateVector]];
  i: CARDINAL;

  FOR i IN [0..LENGTH[Dstate]) DO
    Dstate[i] ← DebugUtilityDefs.MREAD[sp+i]
    ENDLOOP;
  state.instbyte ← b;
  FOR i IN [0..LENGTH[Dstate]) DO
    DebugUtilityDefs.MWRITE[sp+i, Dstate[i]]
    ENDLOOP;
  RETURN
  END;

Break: PUBLIC PROCEDURE [sp: CoreSwapDefs.SVPointer] =
  BEGIN OPEN DebuggerDefs, DebugBreakptDefs;
  -- breakpoint interpreter
  c: CHARACTER;
  bsei: SymDefs.ISEIndex;
  f: ControlDefs.FrameHandle ← DebugUtilityDefs.MREAD[@sp.dest];
  gframe: ControlDefs.GlobalFrameHandle ← DebugUtilityDefs.MREAD[@f.accesslink];
  sfh: SymFrameHandle;
  bp: BBPointer;
  FR: FormatRecord ←
        [indentation: 2, symid: TRUE, firstsym: TRUE, symdelim: '=,
         intersym: IODefs.CR, startchar: IODefs.NUL, termchar: IODefs.NUL];

  IF ~ DebugUtilityDefs.CheckFrame[f] THEN
    BEGIN
    IODefs.WriteString["Breakpoint"L];
    DebugMiscDefs.DebugCommand[sp];
    RETURN
    END;
  IF (bp ← FindBPRec[gframe, FrameRelBPC[f]]) = NIL THEN
    BEGIN
    IF (bp ← FindSegBPRec[gframe, FrameRelBPC[f]]) = NIL THEN
      BEGIN
      IODefs.WriteString[" Breakpoint not found!"L];
      DebugMiscDefs.DebugCommand[sp];
      RETURN
      END;
    BreakInstToState[sp, bp.brkinst];
    RETURN
    END;
  DebugMiscDefs.WriteEOL[];
  BreakInstToState[sp, bp.brkinst];
  IF bp.exo # octal THEN
    BEGIN OPEN DebugSymbolDefs;
    sfh.faddr ← f;
    sfh.stbase ← DAcquireSymbolTable[SymbolsForGFrame[gframe]];
    END;
  IF bp.bt = trace THEN
    BEGIN
    DDptr.StatePtr ← sp;
    bsei ← BodySei[sfh];
    IF bp.exo = entry THEN
      BEGIN
      WriteHerald[sfh,trace,entry];
      DebugMiscDefs.WriteEOL[];
      DumpParameters[bsei, @FR, sfh];
      END ELSE
    IF bp.exo = exit THEN
      BEGIN
      WriteHerald[sfh,trace,exit];
      DebugMiscDefs.WriteEOL[];
      DumpValsFromState[bsei, @FR, sfh.stbase, sp
         !AmIaRecord => RESUME[FALSE]];
      END
    ELSE WriteHerald[sfh,trace,in];
    DO
      IODefs.WriteChar[' ]; IODefs.WriteChar['>];
      c ← IODefs.ReadChar[];
      IODefs.WriteChar[c];
      SELECT c FROM
        'q, 'Q => BEGIN DebugSymbolDefs.DReleaseSymbolTable[sfh.stbase];
                  EXIT; END;
        'p, 'P => BEGIN DumpParameters[bsei, @FR, sfh]; END;
        'v, 'V => BEGIN DumpLocals[bsei, @FR, sfh]; END;
        'r, 'R => BEGIN DumpRetVals[bsei, @FR, sfh]; END;
        'b, 'B => BEGIN DebugSymbolDefs.DReleaseSymbolTable[sfh.stbase];
                  DebugMiscDefs.DebugCommand[sp]; EXIT; END;
        's, 'S => BEGIN DumpSource[bsei, @FR, sfh]; END;
        ENDCASE => BEGIN ListOptions[TRUE]; END;
      ENDLOOP;
    DebugMiscDefs.WriteEOL[]
    END
  ELSE
    BEGIN
    SELECT bp.exo FROM
      entry => WriteHerald[sfh,break,entry];
      exit => WriteHerald[sfh,break,exit];
      octal => WriteOctalHerald[gframe, bp.pc];
      ENDCASE => WriteHerald[sfh,break,in];
    IF bp.exo # octal THEN DebugSymbolDefs.DReleaseSymbolTable[sfh.stbase];
    DebugMiscDefs.DebugCommand[sp];
    END;
  RETURN
  END;

WriteOctalHerald: PUBLIC PROCEDURE [f: ControlDefs.GlobalFrameHandle,
  b: ControlDefs.BytePC] =
  BEGIN OPEN IODefs;
  WriteString["Octal-break in frame: "L];
  WriteOctal[f];
  WriteString[", byte-pc: "L];
  WriteOctal[b];
  RETURN
  END;

WriteHerald: PROCEDURE [f: DebuggerDefs.SymFrameHandle,
  bt: DebugBreakptDefs.BTtype, exi: DebugBreakptDefs.EXOItype] =
  BEGIN
  IODefs.WriteString[IF bt = break THEN "Break"L ELSE "Trace"L];
  IODefs.WriteString[SELECT exi FROM
              entry => " at entry to "L,
              exit => " at exit from "L,
              ENDCASE => " in "L];
  DebuggerDefs.WriteFrameLocus[f, exi = in];
  RETURN
  END;

UCSHandler: PUBLIC PROCEDURE [psv: CoreSwapDefs.SVPointer, signal: UNSPECIFIED] =
  BEGIN OPEN ControlDefs, DebuggerDefs, DebugUtilityDefs;
  sh: SeiHandle;
  FR: FormatRecord ←
       [indentation: 2, symid: TRUE, firstsym: TRUE, symdelim: '=,
        intersym: IODefs.CR, startchar: IODefs.NUL, termchar: IODefs.NUL];
  ucs: PROCEDURE =
    BEGIN IODefs.WriteOctal[signal]; msg[]; RETURN END;
  msg: PROCEDURE =
    BEGIN
    IODefs.WriteString[", msg = "L];
    IODefs.WriteOctal[MREAD[@psv.stk[0]]];
    RETURN
    END;

  BEGIN
  IODefs.WriteString["*** uncaught SIGNAL "L];
  IF signal = -1 THEN GOTO error;
  WriteTransferName[sh ← SigSeiHandle[signal ! ANY => GOTO nosym], TRUE,
    NullFrame, MREAD[@GFT[LOOPHOLE[signal, SignalDesc].gfi].frame]];
  DumpValsFromState[sh.sei, @FR, sh.stbase, psv
       ! AmIaRecord => RESUME[FALSE];
     ANY => GOTO stop];
  DebugSymbolDefs.DReleaseSymbolTable[sh.stbase];
  EXITS
    error => IODefs.WriteString["ERROR"L];
    nosym => ucs[];
    stop => BEGIN IODefs.WriteChar['?]; msg[]; END;
  END;
  DebugMiscDefs.DebugCommand[psv];
  RETURN
  END;

-- initialize the "world"

initialstate: ControlDefs.StateVector;

Install: PUBLIC PROCEDURE =
  BEGIN
-- not worked out yet
  RETURN
  END;

ReInitWindows: PROCEDURE =
  BEGIN OPEN WindowDefs;
  default: WindowHandle = GetCurrentDisplayWindow[];
  default.eofindex ← default.fileindex ← StreamDefs.StreamIndex[0,0];
  default.tempindex ← StreamDefs.StreamIndex[0,-1];
  DDptr.userwindow.fileindex ← StreamDefs.StreamIndex[0,0];
  PaintDisplayWindow[default];
  RETURN
  END;

CopyUserNamePassword: PROCEDURE =
  BEGIN OPEN OsStaticDefs, DebugUtilityDefs;
  userStatics: POINTER TO OsStaticRecord = AREAD[OsStatics];
  debuggerStatics: POINTER TO OsStaticRecord = OsStatics↑;
  pc: TYPE = POINTER TO CARDINAL;
  copy: PROCEDURE [from: POINTER, to: POINTER, nwords: CARDINAL] =
    BEGIN OPEN DebugUtilityDefs;
    WHILE nwords # 0 DO
      to↑ ← AREAD[from]; to ← to + 1; from ← from + 1;
      nwords ← nwords - 1;
      ENDLOOP;
    END;
  copy[from: AREAD[@userStatics.UserName],
       to: debuggerStatics.UserName,
       nwords: LOOPHOLE[debuggerStatics.UserName-1,pc]↑];
  copy[from: AREAD[@userStatics.UserPassword],
       to: debuggerStatics.UserPassword,
       nwords: LOOPHOLE[debuggerStatics.UserPassword-1,pc]↑];
  END;

WriteDebuggerHerald: PROCEDURE =
  BEGIN OPEN IODefs, TimeDefs;
  time: STRING ← [18];
  DebugMiscDefs.WriteEOL[];
  WriteLine[herald];
  AppendDayTime[time,UnpackDT[DefaultTime]];
  time.length ← time.length - 3;
  WriteLine[time];
  IF DDptr.restartmessage # NIL THEN
    BEGIN WriteLine[DDptr.restartmessage]; DDptr.restartmessage ← NIL END;
  WriteChar[CR];
  RETURN
  END;

case: {installing, initial, ucs, cleanmap} ← installing;

-- External Debugger starts here

STOP; -- Restarted when ready to do the coreswap part of install

DO
  BEGIN OPEN DebugUtilityDefs;
  ENABLE
   BEGIN
    InitDebugger =>
      BEGIN OPEN initialstate;
      case ← initial;
      stk[0] ← sp;
      stk[1] ← message;
      GOTO init
      END;
    InitUCSHandler =>
      BEGIN OPEN initialstate;
      case ← ucs;
      stk[0] ← sp;
      stk[1] ← signal;
      GOTO init
      END;
    InitMapCleanup =>
      BEGIN
      case ← cleanmap;
      GOTO init
      END
   END;
  WriteDebuggerHerald[];
  DDptr.gContext ← DDptr.lContext ← DDptr.pContext ← NIL;
  SELECT case FROM
    installing => BEGIN CoreSwap[install]; STOP END;
    initial =>
      BEGIN OPEN initialstate;
      IF stk[1] # NIL THEN UserWriteString[stk[1]];
      DebugMiscDefs.DebugCommand[stk[0]];
      CoreSwap[proceed];
      END;
    ucs =>
      BEGIN OPEN initialstate;
      UCSHandler[stk[0], stk[1]];
      CoreSwap[resume];
      case ← initial
      END;
    cleanmap => CoreSwap[proceed];
    ENDCASE;
  EXITS
   init =>
      BEGIN OPEN DebugSymbolDefs, DebugUtilityDefs;
      DebugMiscDefs.SaveSignallerGF[]; DebugUtilitiesInit[];
      InvalidateFileCache[];
      DebugBreakptDefs.ClearBreakBlocks[]; InitializeDrum[];
      ReInitWindows[];
      IF DDptr.debugPilot THEN
        DebugCacheDefs.ReinitMap[DDptr.ESV.mapLog];
      [] ← PurgeUserSymbols[]; DCheckSymbolItems[];
      CopyUserNamePassword[];
      DDptr.worrybreaks ← FALSE;
      DDptr.caseignoring ← TRUE;
      END;
  END;
  ENDLOOP;

END...
```