```
-- WindowsB.Mesa  Edited by Sandman on May 12, 1978  3:40 PM

DIRECTORY
  ImageDefs: FROM "imagedefs" USING [
    AddCleanupProcedure, AddFileRequest, AllReasons, CleanupItem,
    CleanupProcedure, FileRequest],
  RectangleDefs: FROM "rectangledefs" USING [
    ClearBoxInRectangle, GrayArray, GrayPtr, leftmargin],
  SDDefs: FROM "sddefs" USING [sAddFileRequest, SD],
  SegmentDefs: FROM "segmentdefs" USING [
    Append, DefaultVersion, NewFile, Read, Write],
  StreamDefs: FROM "streamdefs" USING [
    AccessOptions, Append, CloseDiskStream, CreateByteStream, DisplayHandle,
    FileLength, GetDefaultDisplayStream, GetDefaultKey, GetIndex,
    NormalizeIndex, OpenDiskStream, Read, SetIndex, StreamError, StreamHandle,
    StreamIndex, TruncateDiskStream, Write],
  StringDefs: FROM "stringdefs" USING [AppendString],
  SystemDefs: FROM "systemdefs" USING [
    AllocateHeapNode, AllocateHeapString, FreeHeapNode, FreeHeapString],
  WindowDefs: FROM "windowdefs" USING [
    DisplayWindow, FileHandle, NullIndex, OriginIndex, Rptr, Selection,
    WindowHandle, WindowType],
  WindowsA: FROM "windowsa" USING [
    currentwindow, defaultwindow, DisplayFileData, linestarts, maxlines,
    PaintDisplayWindow, SetCurrentDisplayWindow, UndoDataSetup,
    UnlinkDisplayWindow, WriteWindowChar];

DEFINITIONS FROM StreamDefs, RectangleDefs, WindowDefs;

WindowsB: PROGRAM [dfn: STRING]
  IMPORTS ImageDefs, RectangleDefs, SegmentDefs, StreamDefs, StringDefs, SystemDefs, WindowsA
  EXPORTS WindowDefs SHARES WindowDefs, StreamDefs, WindowsA =
  BEGIN OPEN WindowsA;


-- GLOBAL Data

  ControlA: CHARACTER = 1C;
  BS: CHARACTER = 10C;
  CR: CHARACTER = 15C;

-- mouse locations

  xmloc: POINTER = LOOPHOLE[424B];
  ymloc: POINTER = LOOPHOLE[425B];
  xcloc: POINTER = LOOPHOLE[426B];
  ycloc: POINTER = LOOPHOLE[427B];

-- Mesa Display Window Routines

  CreateDisplayWindow: PUBLIC PROCEDURE [type: WindowType,
    rectangle: Rptr, ds: DisplayHandle, ks: StreamHandle, name: STRING]
    RETURNS [WindowHandle] =
    BEGIN
    w: WindowHandle;
    w ← SystemDefs.AllocateHeapNode[SIZE[DisplayWindow]];
    w↑ ←
      DisplayWindow[NIL, type, NIL, NIL, NILProc, rectangle, ds, ks, NIL,,,,];
    AlterWindowType[w, type, name];
    SetCurrentDisplayWindow[w];
    RETURN[w];
    END;

  AlterWindowType: PUBLIC PROCEDURE [
    w: WindowHandle, type: WindowType, name: STRING] =
    BEGIN
    -- first undo all stuff for current type
    SELECT w.type FROM
      clear => NULL; -- window is simply cleared on activation
      random => NULL; -- USERS responsibility to repaint screen
      scratch,
      scriptfile,
      file =>           -- data is a window on file
        BEGIN
        IF w.file # NIL THEN
          BEGIN
```

```
        w.file.destroy[w.file];
        w.file ← NIL;
        END;
      END;
    ENDCASE;
  -- now fixup all stuff for new type
  w.type ← type;
  SELECT type FROM
    clear => NULL; -- window is simply cleared on activation
    random => NULL; -- USERS responsibility to repaint screen
    scratch,
    scriptfile,
    file =>           -- data is a window on file
      IF name # NIL THEN
        SetFileForWindow[w, name];
    ENDCASE;
  -- and set name (if not done already)
  IF (w.name = NIL AND name # NIL) OR w.name # name THEN
    BEGIN
    IF w.name # NIL THEN SystemDefs.FreeHeapString[w.name];
    w.name ← SystemDefs.AllocateHeapString[name.length];
    StringDefs.AppendString[w.name, name];
    END;
  -- set current selection null
  w.selection ← Selection[leftmargin,leftmargin,1,1,NullIndex, NullIndex];
  -- setup Stream options based upon stream existance
  IF w.ds # NIL THEN
    BEGIN
    w.ds.options.NoteLineBreak ← TRUE;
    w.ds.options.NoteScrolling ← TRUE;
    w.ds.put ← WriteWindowChar;
    SELECT type FROM
      clear => NULL;
      random => NULL;
      scratch,
      scriptfile => w.ds.options.StopBottom ← FALSE;
      file => w.ds.options.StopBottom ← TRUE;
      ENDCASE;
    END;
  END;

DestroyDisplayWindow: PUBLIC PROCEDURE [w: WindowHandle] =
  BEGIN -- clear it, unlink it deallocate record space and return
  rectangle: Rptr = w.rectangle;
  clearwords: GrayArray ← [0, 0, 0, 0];
  clear: GrayPtr = @clearwords;
  ClearBoxInRectangle[rectangle, 0, rectangle.cw, 0, rectangle.ch, clear];
  IF w = currentwindow THEN
    BEGIN
    IF w = w.link THEN
      BEGIN
      currentwindow ← NIL;
      UndoDataSetup[w];
      END
    ELSE SetCurrentDisplayWindow[w.link];
    END;
  UnlinkDisplayWindow[w];
  IF w.file # NIL THEN w.file.destroy[w.file];
  SystemDefs.FreeHeapNode[w];
  IF w = defaultwindow THEN defaultwindow ← NIL;
  -- later!! must undo anything done to StreamObject
  END;

OpenDisplayWindows: PUBLIC PROCEDURE =
  BEGIN OPEN StreamDefs;
  -- This guy should set up anything to do with display windows
  -- currently used: switching to/from the extenal debugger
  IF defaultwindow.type = scriptfile THEN
    OpenDiskStream[defaultwindow.file
      ! StreamError =>
        BEGIN
        defaultwindow.eofindex ← GetIndex[defaultwindow.file];
        RESUME
        END];
  -- ensure at end
  SetIndex[defaultwindow.file, defaultwindow.eofindex];
```

```
    END;

  CloseDisplayWindows: PUBLIC PROCEDURE =
    BEGIN
    -- define locals
    file: StreamHandle = defaultwindow.file;
    -- This guy cleans up anything to do with display windows
    -- currently used: switching to/from the extenal debugger
    IF file = NIL THEN RETURN;
    IF defaultwindow=currentwindow AND defaultwindow.tempindex=NullIndex THEN
      defaultwindow.eofindex ← GetIndex[file]
    ELSE SetIndex[file, defaultwindow.eofindex];
    file.put[file,CR];
    THROUGH [0..9] DO file.put[file,'~] ENDLOOP;
    CloseDiskStream[file];
    END;

  GetLineTable: PUBLIC PROCEDURE RETURNS[POINTER] =
    BEGIN
    RETURN[@linestarts];
    END;

  GetCurrentDisplayWindow: PUBLIC PROCEDURE RETURNS [WindowHandle] =
    BEGIN
    RETURN[currentwindow];
    END;

  SetFileForWindow: PUBLIC PROCEDURE [w: WindowHandle, filename: STRING]  =
    BEGIN
    SetFileHandleForWindow[w, NIL, filename];
    END;


  SetFileHandleForWindow: PUBLIC PROCEDURE [
    w: WindowHandle, fileh: FileHandle, filename: STRING]  =
    BEGIN OPEN SegmentDefs;
    -- define locals
    access: AccessOptions;
    -- do file type specific stuff
    SELECT w.type FROM
      scratch,  -- data is maintained in scratch file
      scriptfile =>     -- data is maintained in typescript file
        access ← Read+Write+Append;
      file =>           -- data is a window on file
        access ← Read;
      ENDCASE =>  ERROR;
    -- verify file access is ok
    IF fileh = NIL THEN fileh ← NewFile[filename, access, DefaultVersion];
    -- if already one shit can it (and name too)
    IF w.file # NIL THEN w.file.destroy[w.file];
    -- now create a stream associated with the file
    w.file ← CreateByteStream[fileh,access];
    -- set length based on type
    w.fileindex ← w.eofindex ← OriginIndex;
    SELECT w.type FROM
      scriptfile => NULL;       -- data is maintained in typescript file
      scratch =>        -- data is maintained in scratch file
        IF w # currentwindow THEN CloseDiskStream[w.file];
      file =>           -- data is a window on file
        BEGIN
        w.eofindex ← FileLength[w.file];
        IF w # currentwindow THEN CloseDiskStream[w.file];
        END;
      ENDCASE => ERROR;
    -- assign name and display procedure
    IF w.name # filename THEN
      BEGIN
      IF w.name # NIL THEN SystemDefs.FreeHeapString[w.name];
      w.name ← SystemDefs.AllocateHeapString[filename.length];
      StringDefs.AppendString[w.name, filename];
      END;
    w.tempindex ← NullIndex;
    w.displayproc ← DisplayFileData;
    -- set current selection null
    w.selection ← Selection[leftmargin,leftmargin,1,1,NullIndex, NullIndex];
    END;
```

```
  SetIndexForWindow: PUBLIC PROCEDURE [w: WindowHandle, index: StreamIndex] =
    BEGIN
    -- set fileposition
    SELECT w.type FROM
      scratch, scriptfile, file => w.fileindex ← index;
      ENDCASE;
    -- and paint it if it is the current one
    IF w = currentwindow THEN PaintDisplayWindow[w];
    END;

  SetPositionForWindow: PUBLIC PROCEDURE [w: WindowHandle, pos: CARDINAL] =
    BEGIN
    -- define locals
    fileindex: StreamIndex;
    -- set fileposition
    SELECT w.type FROM
      scratch, scriptfile, file =>
        BEGIN
        fileindex ← NormalizeIndex[StreamIndex[0, pos]];
        SetIndexForWindow[w, fileindex];
        END;
      ENDCASE;
    END;

  NILProc: PROCEDURE [w: WindowHandle] =
    BEGIN
    -- Dummy Display procedure
    END;

-- Mesa Display Window Initialization Routine

  setupdefaultwindow: PROCEDURE =
    BEGIN
    -- Smokey asked me to say this is awful and ugly (JDW)
    i: CARDINAL;
    defaultwindow.file ← CreateByteStream[preopen.file,Read+Write+Append];
    defaultwindow.type ← scriptfile;
    defaultwindow.ds.options.StopBottom ← FALSE;
    defaultwindow.tempindex ← NullIndex;
    defaultwindow.displayproc ← DisplayFileData;
    defaultwindow.fileindex ← defaultwindow.eofindex ← OriginIndex;
    FOR i IN [1..maxlines] DO
      linestarts[i] ← NullIndex;
      ENDLOOP;
    END;

  initwindows: PROCEDURE =
    BEGIN
      ds: DisplayHandle;
    -- create the default window
      ds ← GetDefaultDisplayStream[];
      defaultwindow ← CreateDisplayWindow [
        clear, ds.rectangle, ds, GetDefaultKey[], dfn];
      setupdefaultwindow[];
    END;

  CleanupItem: ImageDefs.CleanupItem ←
    ImageDefs.CleanupItem[link:, mask: ImageDefs.AllReasons, proc: Cleanup];

  Cleanup: ImageDefs.CleanupProcedure =
    BEGIN
    SELECT why FROM
      Finish, Abort, Save =>
        BEGIN
        IF defaultwindow.file = NIL THEN RETURN;
        IF defaultwindow.tempindex # NullIndex THEN
          SetIndex[defaultwindow.file, defaultwindow.eofindex];
        StreamDefs.TruncateDiskStream[defaultwindow.file];
        defaultwindow.file ← NIL;
        IF why = Save THEN
          BEGIN
          preopen.file ← NIL;
          preopen.name ← defaultwindow.name;
          ImageDefs.AddFileRequest[@preopen];
          END;
```

```
        END;
      Restore =>
        BEGIN OPEN SegmentDefs;
        IF preopen.file = NIL THEN
          preopen.file ←
            NewFile[defaultwindow.name,Read+Write+Append,DefaultVersion];
        setupdefaultwindow[];
        SetCurrentDisplayWindow[defaultwindow];
        END;
      OutLd, Checkpoint => CloseDisplayWindows[];
      InLd, Continue => OpenDisplayWindows[];
      Restart =>
        BEGIN OPEN StreamDefs;
        OpenDisplayWindows[ ! StreamError =>
          BEGIN IF error = StreamEnd THEN RESUME END];
        defaultwindow.file.reset[defaultwindow.file];
        END;
      ENDCASE;
    END;

-- MAIN BODY CODE

  preopen: short ImageDefs.FileRequest ← ImageDefs.FileRequest [
    file: NIL, access: Read+Write+Append, link:, body: short[fill:, name: dfn]];

  IF SDDefs.SD[SDDefs.sAddFileRequest] # 0 THEN
    BEGIN
    ImageDefs.AddFileRequest[@preopen];
    STOP;
    END;

  IF preopen.file = NIL THEN
    BEGIN OPEN SegmentDefs;
    preopen.file ← NewFile[dfn,Read+Write+Append,DefaultVersion];
    END;
  initwindows[];
  ImageDefs.AddCleanupProcedure[@CleanupItem];

  END...  of Window
```