```
--File:  WEControl.mesa
--Edited by:
--          Barbara  May 15, 1978  5:58 PM
--          Sandman  April 21, 1978  12:19 PM

DIRECTORY
  ProcessDefs: FROM "processdefs" USING [Detach],
  RectangleDefs: FROM "rectangledefs" USING [
    ComputeCharWidth, GetDefaultBitmap, GetDefaultFont, leftmargin, Rptr,
    xCoord, yCoord],
  StreamDefs: FROM "streamdefs" USING [
    ClearDisplayChar, DisplayHandle, EqualIndex, GetDefaultDisplayStream,
    GetDefaultKey, GetIndex, KeyboardHandle, ModifyIndex, SetIndex,
    StreamError, StreamIndex],
  StringDefs: FROM "stringdefs" USING [AppendChar, AppendString],
  SystemDefs: FROM "systemdefs" USING [AllocateHeapString, FreeHeapString],
  WindExDefs: FROM "windexdefs" USING [
    LoadThisWindow, maxscratch, OriginIndex, WEBreak, WEDataObject, WEMain,
    WEPosition, WESelection, WEWindows, WindowExecutive],
  WindowDefs: FROM "windowdefs" USING [
    AlterWindowType, GetCurrentDisplayWindow, MakeSelection, MarkSelection,
    Selection, WindowHandle];

DEFINITIONS FROM WindExDefs;

WEControl:  PROGRAM
  IMPORTS ProcessDefs, SystemDefs, StringDefs, StreamDefs, RectangleDefs,
    WindowDefs, WindExDefs
  EXPORTS WindExDefs
  SHARES WindExDefs, StreamDefs =
  BEGIN

  -- common types

  WindowHandle: TYPE = WindowDefs.WindowHandle;
  DisplayHandle: TYPE = StreamDefs.DisplayHandle;
  KeyboardHandle: TYPE = StreamDefs.KeyboardHandle;
  StreamIndex: TYPE = StreamDefs.StreamIndex;
  Selection: TYPE = WindowDefs.Selection;
  Rptr: TYPE = RectangleDefs.Rptr;
  xCoord: TYPE = RectangleDefs.xCoord;
  yCoord: TYPE = RectangleDefs.yCoord;

  --

  -- Window Executive Main Control Routine

  ReadEditChar: PUBLIC PROCEDURE [char: CHARACTER, w: WindowHandle] =
    BEGIN
    --declare locals
    index:  StreamIndex;
    fixup:  BOOLEAN ← FALSE;
    firstchar:  BOOLEAN ← TRUE;
    ch:  CHARACTER;
    controlA:  CHARACTER = 1C;
    controlH:  CHARACTER = 10C;
    controlW:  CHARACTER = 27C;
    controlQ:  CHARACTER = 21C;
    ESC: CHARACTER = 33C;
    Space: CHARACTER = 40C;
    --do editing like ReadEditString
    SELECT char FROM
       controlA, controlH =>
         BEGIN
         IF w.ds.charx # RectangleDefs.leftmargin THEN
           BEGIN
           index ← StreamDefs.GetIndex[w.file];
           w.eofindex ← index;
           index ← StreamDefs.ModifyIndex[index, -1];
           IF StreamDefs.EqualIndex[w.selection.rightindex, index] THEN
             BEGIN
             WindowDefs.MarkSelection[w];
             fixup ← TRUE;
             END;
           StreamDefs.SetIndex[w.file, index];
           ch ← w.file.get[w.file];
```

```
          StreamDefs.ClearDisplayChar[w.ds, ch];
          IF fixup THEN
            BEGIN
            w.selection.rightx ← w.ds.charx;
            index ← StreamDefs.ModifyIndex[index, -1];
            w.selection.rightindex ← index;
            WindowDefs.MarkSelection[w];
            fixup ← FALSE;
            END;
          END;
        END;
      controlW, controlQ =>
        BEGIN
        DO
        IF w.ds.charx = RectangleDefs.leftmargin THEN EXIT;
        index ← StreamDefs.GetIndex[w.file];
        index ← StreamDefs.ModifyIndex[index, -1];
        IF StreamDefs.EqualIndex[w.selection.rightindex, index] THEN
          BEGIN
          WindowDefs.MarkSelection[w];
          fixup ← TRUE;
          END;
        StreamDefs.SetIndex[w.file, index];
        w.eofindex ← index;
        ch ← w.file.get[w.file
          ! StreamDefs.StreamError => EXIT];
        IF ch = Space AND NOT firstchar THEN EXIT
          ELSE IF ch # Space THEN firstchar ← FALSE;
        StreamDefs.ClearDisplayChar[w.ds, ch];
        ENDLOOP;
        IF fixup THEN
          BEGIN
          index ← StreamDefs.ModifyIndex[index, -1];
          w.selection.rightindex ← index;
          w.selection.rightx ← w.ds.charx;
          WindowDefs.MarkSelection[w];
          fixup ← FALSE;
          END;
        END;
      ESC => LoadThisWindow[w];
      ENDCASE => MakeOrExtendSelection[w, char];
    END;

MakeOrExtendSelection: PROCEDURE[w: WindowHandle, char: CHARACTER] =
  BEGIN OPEN WEState;
  -- declare locals
  ds: DisplayHandle ← w.ds;
  sel: Selection;
  index: StreamIndex ← StreamDefs.GetIndex[w.file];
  -- now make/extend the current selection
  IF NOT ds.charx = w.selection.rightx OR
    StreamDefs.EqualIndex[OriginIndex, index] THEN
    BEGIN --make this char the current selection
    w.ds.put[w.ds, char];
    sel ← [
      leftx: ds.charx - RectangleDefs.ComputeCharWidth[char, ds.pfont],
      leftline: ds.line,
      leftindex: index,
      rightx: ds.charx ,
      rightline: ds.line,
      rightindex: index
      ];
    END
  ELSE
    BEGIN -- extend it to include this char
    w.ds.put[w.ds, char];
    sel ← Selection[
      leftx: w.selection.leftx,
      leftline: w.selection.leftline,
      leftindex: w.selection.leftindex,
      rightx: ds.charx ,
      rightline: ds.line,
      rightindex: index
      ];
  END;
  WindowDefs.MakeSelection[w, @sel];
  END;

NoteNameError: PUBLIC PROCEDURE [w:WindowHandle, str:  STRING] =
  BEGIN OPEN WEState;
  i: INTEGER;
  scratchstr: STRING;
  -- convert window into scratch and tell bad name
  IF w.type # scratch THEN
    BEGIN
    [scratchstr, i] ← AssignScratchFile[];
    WindowDefs.AlterWindowType[w, scratch, scratchstr];
    scratchfiles[i] ← w.file;
    SystemDefs.FreeHeapString[scratchstr];
    END;
  WriteMessageString[w, str];
  WriteMessageString[w, "FileNameError!"L];
  END;

WriteMessageString: PUBLIC PROCEDURE [w:WindowHandle, str: STRING] =
  BEGIN
  i: CARDINAL;
  -- write message
  FOR i IN [0..str.length) DO
    w.ds.put[w.ds, str[i]];
    ENDLOOP;
  w.ds.put[w.ds, 15B];
  END;

AssignScratchFile: PUBLIC PROCEDURE RETURNS[STRING, INTEGER] =
  BEGIN OPEN WEState;
  zero: CARDINAL = LOOPHOLE['0];
  i: INTEGER;
  str: STRING;
  -- loop through array looking for a free one
  FOR i IN [0..maxscratch) DO
    IF scratchfiles[i] = NIL THEN
      BEGIN
      str ← SystemDefs.AllocateHeapString[8];
      StringDefs.AppendString[str, "Scratch"L];
      StringDefs.AppendChar[str, LOOPHOLE[i+zero,CHARACTER]];
      RETURN[str, i];
      END;
    ENDLOOP;
  END;

-- initialization for wmanager

  InitConfiguration: PROCEDURE =
    BEGIN OPEN WindExDefs;
    START WESelection[@WEState];
    START WEWindows[@WEState];
    START WEPosition[@WEState];
    START WEBreak[@WEState]; -- must be started after WEWindows
    START WEMain[@WEState];
    END;

  InitManager: PROCEDURE =
    BEGIN OPEN WEState;
    -- Declare Locals
    i: CARDINAL;
    w: WindowHandle ← WindowDefs.GetCurrentDisplayWindow[];
    -- process and save currently extant windows
    FOR i IN [0..4) DO
      windows[i] ← w;
      IF w.link = windows[0] THEN EXIT
      ELSE w ← w.link;
      ENDLOOP;
    FOR i IN [0..maxscratch) DO
      scratchfiles[i] ← NIL;
      ENDLOOP;
    -- now init some stuff for later
    defaultmapdata ← RectangleDefs.GetDefaultBitmap[];
    defaultds ← StreamDefs.GetDefaultDisplayStream[];
    defaultks ← StreamDefs.GetDefaultKey[];
    [defaultfont, defaultlineheight] ← RectangleDefs.GetDefaultFont[];
    currentcursor ← textpointer;
    -- setup External Button Procedures
    ButtonProcArray ← TextProcArray;
    ProcessDefs.Detach[FORK WindowExecutive];
    END;

  WEState: WindExDefs.WEDataObject;

-- MAIN BODY CODE

  InitConfiguration[];
  InitManager[];

END. of WEControl
```