```
-- Stats.Mesa  Edited by Sandman on September 1, 1977  6:15 PM

DIRECTORY
  AltoDefs: FROM "altodefs",
  AltoFileDefs: FROM "altofiledefs",
  BcdDefs: FROM "bcddefs",
  ControlDefs: FROM "controldefs",
  CmdDefs: FROM "cmddefs",
  DoubleDefs: FROM "doubledefs",
  FrameDefs: FROM "framedefs",
  IODefs: FROM "iodefs",
  ImageDefs: FROM "imagedefs",
  InlineDefs: FROM "inlinedefs",
  SegmentDefs: FROM "segmentdefs",
  StreamDefs: FROM "streamdefs",
  StringDefs: FROM "stringdefs",
  SymDefs: FROM "symdefs",
  SystemDefs: FROM "systemdefs";

DEFINITIONS FROM AltoDefs, AltoFileDefs, CmdDefs, IODefs, SegmentDefs, StringDefs, StreamDefs;

Stats: PROGRAM
  IMPORTS CmdDefs, DoubleDefs, FrameDefs, IODefs, SegmentDefs, StreamDefs, StringDefs,
    SystemDefs =

  PUBLIC BEGIN

-- types and globals

  stattype: TYPE = {char,line,codebytes,framesize,ngfi,
                    linkbase,nlinks,codepages,sympages,fsi};

  charField,codebyteField: CARDINAL = 7;
  nlinksField,lineField,framesizeField: CARDINAL = 6;
  codepageField,sympageField: CARDINAL = 5;
  linkbaseField,fsiField: CARDINAL = 4;
  ngfiField: CARDINAL = 3;
  filenameField: CARDINAL = 26;

  file: FileHandle ← NIL;
  code: FileSegmentHandle ← NIL;
  syms: FileSegmentHandle ← NIL;
  cmdstream: StreamHandle;
  stream: StreamHandle;
  lc: INTEGER;
  full: INTEGER = 18;
  buffer: POINTER;
  BufSize: CARDINAL = 40*256;

  stats: ARRAY stattype OF CARDINAL ← [0,0,0,0,0,0,0,0,0,0];

  total,subtotal: ARRAY stattype OF DoubleDefs.LongCARDINAL ←
    [[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0]];

  format: ARRAY stattype OF IODefs.NumberFormat =
     [[base: 10, zerofill: FALSE, unsigned: TRUE, columns:charField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:lineField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:codebyteField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:framesizeField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:ngfiField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:linkbaseField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:nlinksField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:codepageField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:sympageField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:fsiField]];

  header: ARRAY stattype OF STRING =
      ["chars ","lines ","codebytes ","framesize ","ngfi ","linkbase ",
        "nlinks ","codepages ","sympages ","fsi "];

 -- the following should be sets
  StatsWanted,localStatsWanted: ARRAY stattype[char..fsi] OF BOOLEAN ←
     [TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE];

  CallDebugger: SIGNAL = CODE;
```

```
-- procedures

  BcdSwitches: PROCEDURE [wanted: BOOLEAN,st:DESCRIPTOR FOR ARRAY stattype OF BOOLEAN] =
    BEGIN
    st[codebytes]←wanted;
    st[framesize]←wanted;
    st[ngfi]←wanted;
    st[linkbase]←wanted;
    st[nlinks]←wanted;
    st[codepages]←wanted;
    st[sympages]←wanted;
    st[fsi]←wanted;
    END;

  GetBcdStats: PROCEDURE [codeseg,symseg:FileSegmentHandle] =
    BEGIN OPEN ControlDefs, SegmentDefs;
    slot: [0..maxallocslot];
    cp: POINTER TO CsegPrefix;
    codebase, codeend: POINTER;
    SwapIn[codeseg];
    cp ← codebase ← FileSegmentAddress[codeseg];
    stats[nlinks] ← cp.nlinks;
    stats[ngfi] ← cp.ngfi;
    stats[linkbase] ← cp.linkbase;
    stats[fsi] ← slot ← cp.EntryVector[MainBodyIndex].framesize;
    stats[framesize] ← IF slot = maxallocslot
         THEN (codebase+cp.EntryVector[MainBodyIndex].initialpc-1)↑
         ELSE FrameDefs.FrameSize[slot];
    codeend ← codebase + InlineDefs.BITSHIFT[codeseg.pages,LogPageSize];
    DO -- until something non-zero
      codeend ← codeend-1;
      IF codeend↑#0 THEN EXIT;
      ENDLOOP;
    stats[codepages] ← codeseg.pages;
    stats[codebytes] ← InlineDefs.BITSHIFT [
      LOOPHOLE[codeend-LOOPHOLE[codebase,CARDINAL]+1],1];
    Unlock[codeseg];
    stats[sympages] ← symseg.pages;
    RETURN
    END;

  GetModule: PROCEDURE [file:FileHandle] RETURNS [codeseg,symseg:FileSegmentHandle] =
    BEGIN
    bcd: POINTER TO BcdDefs.BCD;
    pages: AltoDefs.PageCount;
    mtb: CARDINAL;
    mti: BcdDefs.MTIndex = FIRST[BcdDefs.MTIndex];
    bcdseg: FileSegmentHandle ← NewFileSegment[file,1,1,Read];
    SwapIn[bcdseg];
    bcd ← FileSegmentAddress[bcdseg];
    IF (pages ← bcd.nPages) # 1 THEN
      BEGIN
      Unlock[bcdseg];
      MoveFileSegment[bcdseg, 1, pages];
      SwapIn[bcdseg];
      bcd ← FileSegmentAddress[bcdseg];
      END;
    IF bcd.versionident # BcdDefs.VersionID THEN
      BEGIN
      WriteString["  bad version ID "];
      WriteDecimal[bcd.versionident];
      Unlock[bcdseg];
      DeleteFileSegment[bcdseg];
      RETURN[NIL, NIL]
      END;
    IF bcd.nModules # 1 THEN
      BEGIN
      WriteString["  too many modules: "];
      WriteDecimal[bcd.nModules];
      Unlock[bcdseg];
      DeleteFileSegment[bcdseg];
      RETURN[NIL, NIL]
      END;
    mtb ← LOOPHOLE[bcd,CARDINAL]+bcd.mtOffset;
    codeseg ← IF bcd.definitions THEN NIL ELSE FindSegment[bcdseg, (mtb+mti).cseg, FALSE];
    symseg ← FindSegment[bcdseg, (mtb+mti).sseg, FALSE];
```

```
    Unlock[bcdseg];
    DeleteFileSegment[bcdseg];
    RETURN
    END;

  FindSegment: PROCEDURE [seg: FileSegmentHandle, segdesc: BcdDefs.SegDesc, long: BOOLEAN]
    RETURNS [FileSegmentHandle] =
    BEGIN
    ss: StringDefs.SubStringDescriptor;
    file: SegmentDefs.FileHandle;
    name: STRING;
    bcd: POINTER TO BcdDefs.BCD ← FileSegmentAddress[seg];
    IF segdesc.file = BcdDefs.FTNull THEN RETURN[NIL]
    ELSE IF segdesc.file = BcdDefs.FTSelf THEN file ← seg.file
    ELSE
      BEGIN OPEN f: LOOPHOLE[bcd+bcd.ftOffset, CARDINAL]+segdesc.file;
      name ← SystemDefs.AllocateHeapString[f.name.length+4];
      ss ← [LOOPHOLE[bcd+bcd.ssOffset, STRING], f.name.offset, f.name.length];
      StringDefs.AppendSubString[name, @ss];
      CheckForExtension[name, ".bcd"];
      file ← NewFile[name, DefaultAccess, DefaultVersion];
      SystemDefs.FreeHeapString[name];
      END;
    RETURN[NewFileSegment[file, segdesc.base,
      segdesc.pages + (IF long THEN segdesc.extraPages ELSE 0), Read]];
    END;

  CheckForExtension: PROCEDURE [name, ext: STRING] =
    BEGIN
    i: CARDINAL;
    FOR i IN [0..name.length) DO
      IF name[i] = '. THEN RETURN;
      ENDLOOP;
    StringDefs.AppendString[name, ext];
    RETURN
    END;

  GetSrcStats: PROCEDURE [stream: StreamHandle] =
    BEGIN
    count, i: CARDINAL;
    nc, nl: CARDINAL ← 0;
    crock: STRING = LOOPHOLE[buffer-2];
    UNTIL (count ← ReadBlock[
      stream: stream, address: buffer, words: BufSize !
        StreamError => CONTINUE])=0 DO
      FOR i IN [0..count*2) DO
        IF crock[i] = IODefs.CR THEN nl ← nl+1;
        nc ← nc+1;
        ENDLOOP;
      ENDLOOP;
    stats[char] ← nc; stats[line] ← nl;
    RETURN
    END;

  GetStats: PROCEDURE =
    BEGIN
    name: STRING ← [40];
    switches: STRING ← [10];
    command: BOOLEAN;
    i: CARDINAL ← 0;
    headingWanted: BOOLEAN ← TRUE;

    buffer ← SystemDefs.AllocateSegment[BufSize];
    SetCommandInput[];
    WHILE NextFile[name,switches] DO
     IF name[0] = '? THEN HelpMessage[]
     ELSE
      BEGIN
      localStatsWanted ← StatsWanted; command ← FALSE;
      i ← 0;
      WHILE i < switches.length DO
        SELECT switches[i] FROM
          'b,'B => BcdSwitches[TRUE,DESCRIPTOR[localStatsWanted]];
          'm,'M => SourceSwitches[TRUE,DESCRIPTOR[localStatsWanted]];
          'x,'X=> BEGIN
                  BcdSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
```

```
                ManagerSwitches[TRUE,DESCRIPTOR[localStatsWanted]];
                END;
          '- =>BEGIN
               i ← i+1;
               SELECT switches[i] FROM
                'b,'B => BcdSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
                'm,'M => SourceSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
                'x,'X => ManagerSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
                ENDCASE=> HelpMessage[];
               END;
          'c,'C => BEGIN
                SELECT name[0] FROM
                 'b,'B => BcdSwitches[TRUE,DESCRIPTOR[StatsWanted]];
                 'd,'D => CallDebugger[];
                 'h,'H => Heading[];
                 'm,'M => SourceSwitches[TRUE,DESCRIPTOR[StatsWanted]];
                 'x,'X=> BEGIN
                         BcdSwitches[FALSE,DESCRIPTOR[StatsWanted]];
                         ManagerSwitches[TRUE,DESCRIPTOR[StatsWanted]];
                         END;
                 't,'T => Total["TOTAL:",FALSE];
                 's,'S => Total["SUBTOTAL:",TRUE];
                 '- => SELECT name[1] FROM
                     'b,'B => BcdSwitches[FALSE,DESCRIPTOR[StatsWanted]];
                     'm,'M => SourceSwitches[FALSE,DESCRIPTOR[StatsWanted]];
                     'x,'X => ManagerSwitches[FALSE,DESCRIPTOR[StatsWanted]];
                      ENDCASE=> HelpMessage[];
                 ENDCASE => HelpMessage[];
                command ← TRUE;
                END;
          ENDCASE => HelpMessage[];
        i ← i + 1;
        ENDLOOP;
      IF NOT command THEN
        BEGIN
        IF headingWanted THEN BEGIN headingWanted ← FALSE; Heading[] END;
        StripExtension[name];
        WriteString[name];
        THROUGH [name.length..filenameField) DO WriteChar[' ] ENDLOOP;
        AppendString[name,".mesa"];
        RecordSrcStats[name !FileNameError =>
            BEGIN SourceSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
            CONTINUE END];
        StripExtension[name]; AppendString[name,".bcd"];
        RecordBcdStats[name !FileNameError =>
            BEGIN BcdSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
            CONTINUE END];
        TallyStats[];
        PrintStats[];
        WriteChar[CR];
        END;
      END;
    ENDLOOP;
    SystemDefs.FreeSegment[buffer];
    END;

  Heading: PROCEDURE =
    BEGIN
    type: stattype;
    WriteChar[CR];
    THROUGH [0..filenameField) DO WriteChar[' ] ENDLOOP;
    FOR type IN stattype DO
      IF localStatsWanted[type] THEN WriteString[header[type]] ENDLOOP;
    WriteChar[CR];
    THROUGH [0..filenameField) DO WriteChar[' ] ENDLOOP;
    FOR type IN stattype DO
     IF localStatsWanted[type] THEN
       BEGIN
       THROUGH [0..format[type].columns) DO WriteChar['-]; ENDLOOP;
       THROUGH [0..2) DO WriteChar[' ]; ENDLOOP;
       END;
     ENDLOOP;
    WriteChar[CR];
    WriteChar[CR];
    END;
```

```
  HelpMessage: PROCEDURE =
    BEGIN
    WriteLine["The following commands are available: "];
    WriteLine["    b,B - bcd stats"];
    WriteLine["    d,D - enter debugger"];
    WriteLine["    h,H  - print column headings"];
    WriteLine["    m,M  - source stats"];
    WriteLine["    s,S - subtotal"];
    WriteLine["    t,T - total"];
    WriteLine["    x,X - source characters, source lines, code bytes, and frame size only"];
    WriteLine["b, m, and x are also valid file switches which override the global settings for the file
**. c and C are switches which indicate a command, i.e. d/c is the command to call the debugger. To sup
**press the printing of a set of stats, either locally or globally, precede the switch or command by a
**minus ('-'). Hence -b/c suppresses the printing of all bcd stats, and foo/-b suppresses the printing
**of bcd stats for file foo. The default settings are b and m."];
    END;


  ManagerSwitches: PROCEDURE [wanted: BOOLEAN,st: DESCRIPTOR FOR ARRAY stattype[FIRST[stattype]..LAST[s
**tattype]] OF BOOLEAN] =
    BEGIN
    st[char] ← wanted;
    st[line] ← wanted;
    st[codebytes] ← wanted;
    st[framesize]← wanted;
    END;

  NextFile: PROCEDURE[name,switches: STRING] RETURNS [BOOLEAN] =
    BEGIN
    temp: STRING ← [80];
    IF cmdstream # NIL THEN
      RETURN[ReadCmdStream[cmdstream,name,switches]]
    ELSE
      BEGIN
      WriteString["File: "];
      ReadID[temp];
      WriteChar[CR];
      RETURN[ReadCmdString[temp,name,switches]];
      END;
    END;



  PrintStats: PROCEDURE =
    BEGIN OPEN  IODefs;
    type: stattype;
    FOR type IN [FIRST[stattype]..LAST[stattype]) DO
      IF localStatsWanted[type] THEN
        BEGIN
        WriteNumber[stats[type], format[type]];
        WriteString[", "];
        END
    ENDLOOP;
    IF localStatsWanted[LAST[stattype]] THEN
      WriteNumber[stats[LAST[stattype]], format[LAST[stattype]]]
    END;


  RecordBcdStats: PROCEDURE [name:STRING] =
    BEGIN OPEN StringDefs;
    type: stattype;
    any: BOOLEAN ← FALSE;
    FOR type IN [ngfi..fsi] DO any ← any OR localStatsWanted[type] ENDLOOP;
    any ← any OR localStatsWanted[codebytes] OR localStatsWanted[framesize];
    IF any THEN
      BEGIN
      file ← NewFile[name, Read,OldFileOnly];
      [code,syms] ← GetModule[file];
      IF code # NIL THEN
        BEGIN
        GetBcdStats[code,syms];
        DeleteFileSegment[code];
        END
      ELSE BcdSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
      IF syms # NIL THEN DeleteFileSegment[syms];
      END;
```

```
    RETURN
    END;

  RecordSrcStats: PROCEDURE [name: STRING] =
    BEGIN OPEN StringDefs, SegmentDefs;
    IF localStatsWanted[char] OR localStatsWanted[line] THEN
      BEGIN
      file ← NewFile[name, Read,OldFileOnly];
      stream ← CreateByteStream[file, Read];
      GetSrcStats[stream];
      stream.destroy[stream];
      END;
    RETURN;
    END;

  SetCommandInput: PROCEDURE =
    BEGIN
    cmdstream ← CreateByteStream[NewFile["Com.cm",Read,OldFileOnly],Read];
    SkipExecCommands[cmdstream]; -- skip Mesa.Image
    SkipExecCommands[cmdstream]; -- skip bcd name
    IF cmdstream.endof[cmdstream] THEN
      BEGIN
      cmdstream.destroy[cmdstream];
      cmdstream ← NIL;
      END;
    END;

  StripExtension: PROCEDURE [name:STRING] =
    BEGIN
    i: CARDINAL;
    FOR i IN [0..name.length) DO
      IF name[i] = '. THEN BEGIN name.length ← i; RETURN END;
    ENDLOOP;
    RETURN
    END;

  SourceSwitches: PROCEDURE[wanted: BOOLEAN, st: DESCRIPTOR FOR ARRAY stattype[FIRST[stattype]..LAST[st
**attype]] OF BOOLEAN] =
    BEGIN
    st[char]←wanted;
    st[line]←wanted;
    END;

  TallyStats: PROCEDURE =
    BEGIN OPEN DoubleDefs;
    type: stattype;
    -- no total for fsil
    FOR type IN [FIRST[stattype]..sympages] DO
      IF localStatsWanted[type] THEN
         subtotal[type] ← DAdd[subtotal[type],
                       LongCARDINAL[highbits:0,lowbits:stats[type]]];
    ENDLOOP;
    END;

  Total: PROCEDURE [name: STRING, subt: BOOLEAN] =
    BEGIN OPEN DoubleDefs;
    type: stattype;
    THROUGH [0..filenameField) DO WriteChar[' ] ENDLOOP;
    FOR type IN stattype DO
      IF localStatsWanted[type] THEN
       BEGIN
       THROUGH [0..format[type].columns) DO WriteChar['-]; ENDLOOP;
       THROUGH [0..2) DO WriteChar[' ]; ENDLOOP;
       END;
     ENDLOOP;
    WriteChar[CR]; WriteChar[CR];
    WriteString[name];
    THROUGH [name.length..filenameField) DO WriteChar[' ] ENDLOOP;
    -- no total for fsil
    FOR type IN [FIRST[stattype]..sympages) DO
      IF localStatsWanted[type] THEN
        BEGIN
        WriteDouble[IF subt THEN subtotal[type] ELSE
                    DAdd[subtotal[type], total[type]],
                    format[type]];
        WriteString[", "];
```

```
        END;
      ENDLOOP;
    IF localStatsWanted[sympages] THEN
      WriteDouble[IF subt THEN subtotal[sympages] ELSE
                  DAdd[subtotal[sympages], total[sympages]],
                  format[sympages]];
    WriteChar[CR]; WriteChar[CR];
    IF subt THEN
      BEGIN
      FOR type IN stattype DO
        total[type] ← DAdd[total[type],subtotal[type]];
        ENDLOOP;
      subtotal ← [[0,0],[0,0],[0,0],[0,0],[0,0],
                  [0,0],[0,0],[0,0],[0,0],[0,0]];
      END;
    END;


WriteDouble: PROCEDURE [num: DoubleDefs.LongCARDINAL, format:IODefs.NumberFormat] =
  BEGIN
  i: CARDINAL;
  temp: STRING ← [20];
  DoubleDefs.AppendDouble[temp,num];
  IF temp.length > format.columns THEN
     FOR i IN [0..format.columns) DO WriteChar['*] ENDLOOP
  ELSE
    BEGIN
    FOR i IN [temp.length..format.columns) DO
      WriteChar[' ];
      ENDLOOP;
    WriteString[temp];
    END;
  END;

-- main body

GetStats[];
ImageDefs.StopMesa[];

END.
```