```
-- Statistics.Mesa  Edited by Sandman on April 20, 1978  9:45 AM

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

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

Statistics: PROGRAM
  IMPORTS IODefs, SegmentDefs, StreamDefs, StringDefs,
    SystemDefs, TimeDefs, MiscDefs =

  PUBLIC BEGIN

-- types and globals

  StatType: TYPE =
    {char,line,codebytes,framesize,ngfi,nlinks,codepages,sympages};

  charField,codebyteField: CARDINAL = 7;
  nlinksField,lineField,framesizeField: CARDINAL = 6;
  codepageField: CARDINAL = 5;
  sympageField: CARDINAL = 4;
  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];

  total,subtotal: ARRAY StatType OF LONG INTEGER ← [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:nlinksField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:codepageField],
     [base: 10, zerofill: FALSE, unsigned: TRUE, columns:sympageField]];

 -- the following should be sets
  StatsWanted,localStatsWanted: PACKED ARRAY StatType OF BOOLEAN ←
     [TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE];

  StatsDesired: TYPE = DESCRIPTOR FOR PACKED ARRAY StatType OF BOOLEAN;

-- procedures

  BcdSwitches: PROCEDURE [wanted: BOOLEAN, st: StatsDesired] =
    BEGIN
    st[codebytes]←wanted;
    st[framesize]←wanted;
```

```
    st[ngfi]←wanted;
    st[nlinks]←wanted;
    st[codepages]←wanted;
    st[sympages]←wanted;
    END;

  GetBcdStats: PROCEDURE [codeseg,symseg:FileSegmentHandle] =
    BEGIN OPEN ControlDefs, SegmentDefs;
    codebase: POINTER;
    cp: POINTER TO CSegPrefix;
    code: POINTER TO PACKED ARRAY [0..0) OF AltoDefs.BYTE;
    codebytes: CARDINAL ← codeseg.pages*AltoDefs.BytesPerPage-1;
    ep: POINTER TO EntryVectorItem;
    SwapIn[codeseg];
    cp ← code ← codebase ← FileSegmentAddress[codeseg];
    stats[nlinks] ← cp.nlinks;
    stats[ngfi] ← cp.ngfi;
    ep ← @cp.entry[MainBodyIndex];
    stats[framesize] ← (codebase+CARDINAL[ep.initialpc] -
      (IF ep.framesize = MaxAllocSlot THEN 2 ELSE 1))↑;
    UNTIL code[codebytes] # 0 DO codebytes ← codebytes-1 ENDLOOP;
    stats[codepages] ← codeseg.pages;
    stats[codebytes] ← codebytes+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 "L];
      WriteDecimal[bcd.versionident];
      Unlock[bcdseg];
      DeleteFileSegment[bcdseg];
      RETURN[NIL, NIL]
      END;
    IF bcd.nModules # 1 THEN
      BEGIN
      WriteString["  too many modules: "L];
      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).code.sgi, FALSE];
    symseg ← FindSegment[bcdseg, (mtb+mti).sseg, FALSE];
    Unlock[bcdseg];
    DeleteFileSegment[bcdseg];
    RETURN
    END;

  FindSegment: PROCEDURE [
    seg: FileSegmentHandle, sgi: BcdDefs.SGIndex, long: BOOLEAN]
    RETURNS [FileSegmentHandle] =
    BEGIN OPEN BcdDefs;
    file: SegmentDefs.FileHandle;
    bcd: POINTER TO BcdDefs.BCD ← FileSegmentAddress[seg];
    ssb: BcdDefs.NameString = LOOPHOLE[bcd+bcd.ssOffset];
```

```
    sgh: SGHandle = LOOPHOLE[bcd+bcd.sgOffset, CARDINAL]+sgi;
    IF sgh.file = BcdDefs.FTNull THEN RETURN[NIL]
    ELSE IF sgh.file = BcdDefs.FTSelf THEN file ← seg.file
    ELSE
      BEGIN
      fth: FTHandle = LOOPHOLE[bcd+bcd.ftOffset, CARDINAL]+sgh.file;
      name: STRING ← SystemDefs.AllocateHeapString[ssb.size[fth.name]+4];
      ss: StringDefs.SubStringDescriptor ←
        [@ssb.string, fth.name, ssb.size[fth.name]];
      StringDefs.AppendSubString[name, @ss];
      CheckForExtension[name, ".bcd"L];
      file ← NewFile[name, DefaultAccess, DefaultVersion];
      SystemDefs.FreeHeapString[name];
      END;
    RETURN[NewFileSegment[file, sgh.base,
      sgh.pages + (IF long THEN sgh.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: POINTER TO PACKED ARRAY OF CHARACTER = buffer;
    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 => MiscDefs.CallDebugger[NIL];
                 '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:"L,FALSE];
                 's,'S => Total["SUBTOTAL:"L,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"L];
        RecordSrcStats[name ! FileNameError =>
          BEGIN
          SourceSwitches[FALSE,DESCRIPTOR[localStatsWanted]];
          CONTINUE
          END];
        StripExtension[name]; AppendString[name,".bcd"L];
        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;
    header: ARRAY StatType OF STRING =
      ["chars "L,"lines "L,"codebytes "L,"framesize "L,"ngfi "L,
        "nlinks "L,"codepages "L,"sympages "L];

    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
       IF type = LAST[StatType] THEN WriteChar[' ];
       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: "L];
    WriteLine["    b,B - bcd stats"L];
    WriteLine["    d,D - enter debugger"L];
    WriteLine["    h,H  - print column headings"L];
    WriteLine["    m,M  - source stats"L];
    WriteLine["    s,S - subtotal"L];
    WriteLine["    t,T - total"L];
    WriteLine["    x,X - source characters, source lines, code bytes, and frame size only"L];
    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."L];
    END;


  ManagerSwitches: PROCEDURE [wanted: BOOLEAN, st:  StatsDesired] =
    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];
    i: CARDINAL ← 0;
    get1: PROCEDURE RETURNS [CHARACTER] =
      BEGIN
      RETURN[IF cmdstream.endof[cmdstream] THEN IODefs.NUL
        ELSE cmdstream.get[cmdstream]]
      END;
    get2: PROCEDURE RETURNS [c: CHARACTER] =
      BEGIN
      IF i < temp.length THEN BEGIN c ← temp[i]; i ← i+1 END
      ELSE c ← IODefs.NUL;
      RETURN[c]
      END;
    IF cmdstream # NIL THEN GetToken[get1,name,switches]
    ELSE
      BEGIN
      WriteString["File: "L];
      ReadID[temp];
      WriteChar[CR];
      GetToken[get2,name,switches];
      END;
    RETURN[name.length # 0];
    END;


  GetToken: PROCEDURE [
    get: PROCEDURE RETURNS [CHARACTER], token, switches: STRING] =
    BEGIN OPEN IODefs;
    s: STRING;
    c: CHARACTER;
    token.length ← switches.length ← 0;
    s ← token;
    WHILE (c ← get[]) # NUL DO
      SELECT c FROM
        SP, CR =>
          IF token.length # 0 OR switches.length # 0 THEN RETURN;
        '/ => s ← switches;
        ENDCASE => StringDefs.AppendChar[s, c];
      ENDLOOP;
    RETURN
    END;


  PrintStats: PROCEDURE =
    BEGIN OPEN  IODefs;
    type: StatType;
    FOR type IN StatType DO
      IF localStatsWanted[type] THEN
```

```
      BEGIN
      WriteNumber[stats[type], format[type]];
      IF type = codepages THEN
        BEGIN
        bytes: CARDINAL ← stats[codebytes]+stats[nlinks]*2;
        bytes ← (IF stats[nlinks] MOD 2 = 0 THEN 4 ELSE 2) + bytes;
        IF SystemDefs.PagesForWords[bytes/AltoDefs.BytesPerWord] #
          stats[codepages] THEN WriteChar['*] ELSE WriteChar[SP];
        END;
      IF type # LAST[StatType] THEN WriteString["  "L];
      END
    ENDLOOP;
    END;


  RecordBcdStats: PROCEDURE [name:STRING] =
    BEGIN OPEN StringDefs;
    type: StatType;
    any: BOOLEAN ← FALSE;
    FOR type IN [ngfi..sympages] 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 OPEN SegmentDefs;
    cfa: POINTER TO AltoFileDefs.CFA ← MiscDefs.CommandLineCFA[];
    IF cfa.fp = AltoFileDefs.NullFP THEN BEGIN cmdstream ← NIL; RETURN END;
    cmdstream ← StreamDefs.CreateByteStream[InsertFile[@cfa.fp, Read], Read
      ! InvalidFP => GOTO noCommandLine];
    StreamDefs.JumpToFA[cmdstream, @cfa.fa ! ANY => GOTO noCommandLine];
    IF cmdstream.endof[cmdstream] THEN
      BEGIN
      cmdstream.destroy[cmdstream];
      cmdstream ← NIL;
      END;
    EXITS
      noCommandLine => cmdstream ← NIL;
    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: StatsDesired] =
    BEGIN
    st[char]←wanted;
```

```
    st[line]←wanted;
    END;

  TallyStats: PROCEDURE =
    BEGIN
    type: StatType;
    FOR type IN StatType DO
      IF localStatsWanted[type] THEN
        subtotal[type] ← subtotal[type]+stats[type];
    ENDLOOP;
    END;

  Total: PROCEDURE [name: STRING, subt: BOOLEAN] =
    BEGIN
    type: StatType;
    THROUGH [0..filenameField) DO WriteChar[' ] ENDLOOP;
    FOR type IN StatType DO
      IF localStatsWanted[type] THEN
       BEGIN
       IF type = LAST[StatType] THEN WriteChar[' ];
       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;
    FOR type IN StatType DO
      IF localStatsWanted[type] THEN
        BEGIN
        WriteDouble[
          IF subt THEN subtotal[type] ELSE subtotal[type]+total[type],
          format[type]];
        IF type # LAST[StatType] THEN WriteString["  "];
        END;
      ENDLOOP;
    WriteChar[CR]; WriteChar[CR];
    IF subt THEN
      BEGIN
      FOR type IN StatType DO
        total[type] ← total[type]+subtotal[type];
        ENDLOOP;
      subtotal ← [0, 0, 0, 0, 0, 0, 0, 0];
      END;
    END;


WriteDouble: PROCEDURE [num: LONG INTEGER, format:IODefs.NumberFormat] =
  BEGIN
  i: CARDINAL;
  temp: STRING ← [20];
  StringDefs.AppendLongNumber[temp,num,10];
  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

time: STRING ← [18];
TimeDefs.AppendDayTime[time, TimeDefs.UnpackDT[TimeDefs.CurrentDayTime[]]];
IODefs.WriteString["

Alto/Mesa Statistics Package

Statistics as of "];
IODefs.WriteLine[time];
GetStats[];
ImageDefs.StopMesa[];
```

```
END.
```