```
-- Version.Mesa  Edited by Wick on October 8, 1977  6:48 PM

DIRECTORY
  AltoFileDefs: FROM "altofiledefs",
  BcdDefs: FROM "bcddefs",
  DirectoryDefs: FROM "directorydefs",
  ImageDefs: FROM "imagedefs",
  IODefs: FROM "iodefs",
  SegmentDefs: FROM "segmentdefs",
  StreamDefs: FROM "streamdefs",
  StringDefs: FROM "StringDefs",
  TimeDefs: FROM "timedefs";

DEFINITIONS FROM IODefs, StreamDefs, StringDefs;

Version: PROGRAM
  IMPORTS DirectoryDefs, IODefs, SegmentDefs, StreamDefs, StringDefs, TimeDefs
  SHARES ImageDefs =
  BEGIN

  FP: TYPE = AltoFileDefs.FP;
  BcdBase: TYPE = POINTER TO BcdDefs.BCD;
  UnpackedTime: TYPE = TimeDefs.UnpackedTime;


  BinaryVersion: PROCEDURE [info: POINTER TO FileInfo] =
    BEGIN OPEN SegmentDefs;
    bcd: BcdBase;
    seg: FileSegmentHandle ←
      NewFileSegment[InsertFile[@info.fp,Read],1,1,Read];
    SwapIn[seg];  bcd ← FileSegmentAddress[seg];
    SELECT bcd.versionident FROM
      BcdDefs.VersionID =>
        BEGIN
        PrintVersion[bcd.version];
        WriteChar[CR];
        IF info.type=object AND verbose THEN
          BEGIN
          IF bcd.nPages > 1 THEN
            BEGIN
            Unlock[seg];
            MoveFileSegment[seg,1,bcd.nPages];
            SwapIn[seg];
            bcd ← FileSegmentAddress[seg];
            END;
          PrintFileVersions[bcd];
          END;
        END;
      ImageDefs.VersionID =>
        BEGIN
        PrintVersion[bcd.version];
        WriteChar[CR];
        END;
      ENDCASE =>
        BEGIN
        WriteString[" bad version ID "];
        WriteDecimal[bcd.versionident];
        WriteChar[CR];
        END;
    Unlock[seg];
    DeleteFileSegment[seg];
    RETURN
    END;

  SourceDate: PROCEDURE [info: POINTER TO FileInfo]  =
    BEGIN
    date: TimeDefs.UnpackedTime ← [0,0,0,0,0,0,FALSE];
    index: CARDINAL[0..12);
    stream: StreamHandle;
    token: STRING ← [40];
    found: BOOLEAN;
    stream ← CreateByteStream[
      SegmentDefs.InsertFile[@info.fp,Read],Read];
    UNTIL GetIndex[stream].page # 0 DO
      ENABLE StreamError => EXIT;
      NextItem[stream,token];
```

```
      [found,index] ← IsMonth[token];
      IF found THEN
        BEGIN
        NextItem[stream,token];
        IF IsNumber[token] THEN
          BEGIN
          date.month  ← index;
          date.day ← StringToDecimal[token];
          NextItem[stream,token];
          IF IsNumber[token] THEN
            date.year ← StringToDecimal[token];
          NextItem[stream,token];
          IF IsNumber[token] THEN
            BEGIN
            date.hour ← StringToDecimal[token];
            NextItem[stream,token];
            IF IsNumber[token] THEN
              date.minute ← StringToDecimal[token];
            NextItem[stream,token];
            IF EquivalentString[token,"PM"]
            AND date.hour < 12 THEN
              date.hour ← date.hour+12;
            END;
          WriteChar[' ']; PrintDate[date];
          WriteChar[CR]; EXIT;
          END;
        END;
      ENDLOOP;
    stream.destroy[stream];
    RETURN
    END;

  Months: ARRAY [0..12) OF STRING = [
    "January","February","March","April","May","June","July",
    "August","September","October","November","December"];

  IsMonth: PROCEDURE [candidate: STRING] RETURNS [BOOLEAN,CARDINAL] =
    BEGIN
    i: CARDINAL[0..12);
    FOR i IN [0..12) DO
      IF EquivalentString[candidate,Months[i]] THEN RETURN[TRUE,i];
      ENDLOOP;
    RETURN[FALSE,0]
    END;

  IsNumber: PROCEDURE [s: STRING] RETURNS [BOOLEAN] =
    BEGIN
    i: CARDINAL;
    FOR i IN [0..s.length) DO
      SELECT s[i] FROM
        IN ['0..'9] => NULL;
        ENDCASE => RETURN[FALSE];
      ENDLOOP;
    RETURN[TRUE]
    END;

  NextItem: PROCEDURE [stream: StreamHandle, token: STRING] =
    BEGIN
    c: CHARACTER;
    token.length ← 0;
    WHILE ~stream.endof[stream] DO
       c ← stream.get[stream];
       SELECT c FROM
         IN ['a..'z], IN ['A..'Z], IN ['0..'9] => AppendChar[token,c];
         ENDCASE => IF token.length # 0 THEN EXIT;
       ENDLOOP;
    RETURN
    END;

  PrintFileVersions: PROCEDURE [bcd: BcdBase] =
    BEGIN OPEN BcdDefs;
    line: STRING ← [40];
    filename: SubStringDescriptor;
    fti: FTIndex;
    ftb: CARDINAL = LOOPHOLE[bcd+bcd.ftOffset];
    stb: STRING = LOOPHOLE[bcd+bcd.ssOffset];
```

```
    FOR fti ← FIRST[FTIndex], fti+SIZE[FTRecord]
    UNTIL fti = bcd.ftLimit DO
      OPEN f: ftb + fti;
      filename ← SubStringDescriptor[
        stb, f.name.offset, f.name.length];
      line.length ← 0;  WriteChar[' ];
      PrintVersion[f.version]; WriteChar[' ];
      AppendSubString[line,@filename];
      WriteLine[line];
      ENDLOOP;
    RETURN
    END;

  PrintVersion: PROCEDURE [stamp: BcdDefs.VersionStamp] =
    BEGIN
    WriteChar[IF stamp.zapped THEN '* ELSE ' ];
    PrintDate[TimeDefs.UnpackDT[stamp.time]];
    WriteChar[' ];
    WriteOctal[stamp.net];
    WriteChar['#];
    WriteOctal[stamp.host];
    WriteChar['#];
    END;

  PrintDate: PROCEDURE [dt: UnpackedTime] =
    BEGIN
    tmp: STRING ← [40];
    TimeDefs.AppendDayTime[tmp,dt];
    WriteString[tmp];
    END;


  -- command line stuff

  NextFile: PROCEDURE[token: STRING] RETURNS [BOOLEAN];

  ReadFromCmdFile: PROCEDURE[token: STRING] RETURNS [BOOLEAN] =
    BEGIN
    RETURN[ReadCmdStream[cmdstream,token]]
    END;

  ReadFromKeyboard: PROCEDURE[token: STRING] RETURNS [BOOLEAN] =
    BEGIN
    WriteString["File: "];
    ReadID[token];  WriteChar[CR];
    RETURN[token.length#0];
    END;

  ReadCmdStream: PROCEDURE [
    stream: StreamHandle, token: STRING]
    RETURNS [BOOLEAN] =
    BEGIN
    c: CHARACTER;
    token.length ← 0;
    WHILE ~stream.endof[stream] DO
       c ← stream.get[stream];
       SELECT c FROM
         SP => IF token.length # 0 THEN EXIT;
         CR => EXIT;
         ENDCASE => AppendChar[token,c];
       ENDLOOP;
    RETURN[token.length#0];
    END;

  cmdstream: StreamHandle;

  SetCommandInput: PROCEDURE =
    BEGIN
    cmdstream ← NewByteStream["Com.Cm",Read];
    [] ← ReadCmdStream[cmdstream, name];
    StripSwitches[name];  -- mesa.image
    [] ← ReadCmdStream[cmdstream, name];
    IF interactive ← cmdstream.endof[cmdstream] THEN
      BEGIN
      cmdstream.destroy[cmdstream];
      NextFile ← ReadFromKeyboard;
```

```
    END
  ELSE NextFile ← ReadFromCmdFile;
  RETURN
  END;

StripSwitches: PROCEDURE [token: STRING] =
  BEGIN
  i, j: CARDINAL;
  option: BOOLEAN ← TRUE;
  FOR i IN [0..token.length) DO
    IF token[i] = '/ THEN
      BEGIN
      FOR j IN (i..token.length) DO
        SELECT token[j] FROM
          '~, '- => option ← ~option;
          'p, 'P => BEGIN pause ← option; option ← TRUE END;
          'v, 'V => BEGIN verbose ← option; option ← TRUE END;
          ENDCASE;
        ENDLOOP;
      token.length ← i;
      END;
    ENDLOOP;
  RETURN
  END;

StripExtension: PROCEDURE [name, ext: STRING] =
  BEGIN
  i, j: CARDINAL;
  IF ext # NIL THEN ext.length ← 0;
  i ← (j ← name.length) - 1;
  IF name[i]='. THEN i ← (j ← i) - 1;
  FOR i ← i, i-1 UNTIL name[i] = '. DO
    IF name[i] = '! THEN j ← i;
    IF i = 0 THEN RETURN;
    ENDLOOP;
  name.length ← i;
  IF ext # NIL THEN
  UNTIL (i←i+1)=j DO
    AppendChar[ext,name[i]];
    ENDLOOP;
  RETURN
  END;

GetFiles: PROCEDURE [fp: POINTER TO FP, fn: STRING] RETURNS [BOOLEAN] =
  BEGIN
  i: FileType;
  ext: STRING ← [40];
  StripExtension[fn,ext];
  IF EquivalentString[fn,name] THEN
    FOR i IN FileType DO
      IF EquivalentString[ext,files[i].ext] THEN
        BEGIN files[i].fp ← fp↑;
        files[i].found ← TRUE; EXIT;
        END;
    ENDLOOP;
  RETURN [FALSE]
  END;


-- main program

FileType: TYPE = {source, config, object, symbols, code, image};

FileInfo: TYPE = RECORD [
  type: FileType,
  found: BOOLEAN,
  fp: FP,
  ext: STRING,
  proc: PROCEDURE[POINTER TO FileInfo]];

files: ARRAY FileType OF FileInfo ← [
  FileInfo[source,,,"mesa",SourceDate],
  FileInfo[config,,,"config",SourceDate],
  FileInfo[object,,,"bcd",BinaryVersion],
  FileInfo[symbols,,,"symbols",BinaryVersion],
  FileInfo[code,,,"code",BinaryVersion],
```

```
    FileInfo[image,,,"image",BinaryVersion]];

  i: FileType;
  name: STRING ← [100];

  interactive: BOOLEAN;
  pause, pauseDflt: BOOLEAN ← FALSE;
  verbose, verboseDflt: BOOLEAN ← FALSE;

  SetCommandInput[];
  pauseDflt ← pause;
  verboseDflt ← verbose;
  WriteChar[CR];
  WHILE NextFile[name] DO
    pause ← pauseDflt;
    verbose ← verboseDflt;
    StripSwitches[name];
    IF name.length = 0 THEN
      BEGIN
      pauseDflt ← pause;
      verboseDflt ← verbose;
      IF interactive THEN WriteChar[CR];
      END
    ELSE
      BEGIN
      FOR i IN FileType DO files[i].found ← FALSE ENDLOOP;
      StripExtension[name, NIL];
      DirectoryDefs.EnumerateDirectory[GetFiles];
      IF ~interactive THEN WriteLine[name];
      FOR i IN FileType DO
        IF files[i].found THEN
          BEGIN
          WriteChar[' ];
          WriteString[files[i].ext];
          WriteChar[':];
          files[i].proc[@files[i]];
          END;
        ENDLOOP;
      WriteChar[CR];
      IF pause THEN [] ← ReadChar[];
      END;
  ENDLOOP;
ImageDefs.StopMesa[];

END.
```