```
-- BcdFileLookup.Mesa; edited by Johnsson on April 13, 1978  8:06 AM

DIRECTORY
  AltoFileDefs: FROM "altofiledefs",
  BcdControlDefs: FROM "bcdcontroldefs",
  BcdDefs: FROM "bcddefs",
  BcdFileDefs: FROM "bcdfiledefs",
  BcdTabDefs: FROM "bcdtabdefs",
  BcdUtilDefs: FROM "bcdutildefs",
  DirectoryDefs: FROM "directorydefs",
  MiscDefs: FROM "miscdefs",
  SegmentDefs: FROM "segmentdefs",
  StringDefs: FROM "stringdefs",
  SystemDefs: FROM "systemdefs",
  TableDefs: FROM "tabledefs";

BcdFileLookup: PROGRAM
  IMPORTS BcdTabDefs, BcdUtilDefs, DirectoryDefs, MiscDefs, SegmentDefs, StringDefs, SystemDefs, TableD
**efs
  EXPORTS BcdFileDefs, BcdControlDefs =
  BEGIN

  FP: TYPE = AltoFileDefs.FP;
  SubString: TYPE = StringDefs.SubString;
  SubStringDescriptor: TYPE = StringDefs.SubStringDescriptor;

  fileArray: DESCRIPTOR FOR ARRAY OF SegmentDefs.FileHandle;
  maxFiles: CARDINAL = 256;

  ftb, stb: TableDefs.TableBase;

  Notifier: TableDefs.TableNotifier =
    BEGIN OPEN BcdDefs;
    ftb ← base[fttype];
    stb ← base[sttype];
    END;

  LookupFileTable: PUBLIC PROCEDURE =
    BEGIN OPEN BcdDefs;
    ss1: SubStringDescriptor;
    i: CARDINAL;
    bcd: SubString = @ss1;
    ftb: TableDefs.TableBase;
    ftLimit: FTIndex;
    filesToFind: CARDINAL;

    checkone: PROCEDURE [fp: POINTER TO FP, name: STRING] RETURNS [BOOLEAN] =
      BEGIN
      i: CARDINAL;
      dirName: SubStringDescriptor;
      fti: FTIndex;
      found: BOOLEAN ← FALSE;
      hti: BcdTabDefs.HTIndex;
      n: NameRecord;
      dirName.base ← name;
      FOR i IN [0..name.length) DO
        IF name[i] = '. THEN
          BEGIN
          IF name.length-i # 5 THEN EXIT;
          dirName.offset ← i+1; dirName.length ← 3;
          IF ~StringDefs.EquivalentSubStrings[@dirName, bcd] THEN EXIT;
          dirName.offset ← 0; dirName.length ← i;
          [found, hti] ← BcdTabDefs.FindEquivalentString[@dirName];
          EXIT
          END;
        ENDLOOP;
      IF ~found THEN
        BEGIN
        dirName.offset ← 0;
        dirName.length ← name.length-1;  -- remove '.
        [found, hti] ← BcdTabDefs.FindEquivalentString[@dirName];
        END;

      IF ~found THEN RETURN[FALSE];
      n ← BcdUtilDefs.NameForHti[hti];
      i ← 0;
```

```
    FOR fti ← FIRST[FTIndex], fti+SIZE[FTRecord] UNTIL fti = ftLimit DO
      IF fileArray[i] = NIL THEN
        BEGIN
        IF (ftb+fti).name = n THEN
          BEGIN OPEN SegmentDefs;
          LockFile[fileArray[i] ← InsertFile[fp, Read]];
          filesToFind ← filesToFind - 1;
          EXIT;
          END;
        END;
      i ← i + 1;
      ENDLOOP;
    RETURN[filesToFind=0]
    END;

  bcd↑ ← [base: "bcd"L, offset: 0, length: 3];
  [ftb,LOOPHOLE[ftLimit,CARDINAL]] ← TableDefs.TableBounds[fttype];
  filesToFind ← LOOPHOLE[ftLimit,CARDINAL]/SIZE[FTRecord];
  FOR i IN [0..LENGTH[fileArray]) DO
    IF fileArray[i] # NIL THEN filesToFind ← filesToFind - 1;
    ENDLOOP;
  DirectoryDefs.EnumerateDirectory[checkone];
  END;

UnknownFile: PUBLIC ERROR [fti: BcdDefs.FTIndex] = CODE;

HandleForFile: PUBLIC PROCEDURE [fti: BcdDefs.FTIndex]
  RETURNS [file: SegmentDefs.FileHandle] =
  BEGIN
  index: CARDINAL = LOOPHOLE[fti,CARDINAL]/SIZE[BcdDefs.FTRecord];
  IF index >= LENGTH[fileArray] OR (file ← fileArray[index]) = NIL THEN
    ERROR UnknownFile[fti];
  RETURN
  END;

AddFile: PROCEDURE [hti: BcdTabDefs.HTIndex] RETURNS [fti: BcdDefs.FTIndex] =
  BEGIN OPEN BcdDefs;
  ftLimit: FTIndex = LOOPHOLE[TableDefs.TableBounds[fttype].size];
  name: NameRecord ← BcdUtilDefs.NameForHti[hti];
  FOR fti ← FIRST[FTIndex], fti+SIZE[FTRecord] UNTIL fti=ftLimit DO
    IF (ftb+fti).name = name THEN RETURN;
    ENDLOOP;
  fti ← TableDefs.Allocate[fttype,SIZE[FTRecord]];
  (ftb+fti)↑ ← [name: name,
    version: [time: [0,0], zapped: FALSE, net: 0, host: 0]];
  RETURN
  END;

BuildFileTable: PUBLIC PROCEDURE =
  BEGIN OPEN BcdDefs, BcdTabDefs;
  sti, stLimit: STIndex;
  fti: FTIndex;
  TableDefs.AddNotify[Notifier];
  stLimit ← LOOPHOLE[TableDefs.TableBounds[sttype].size];
  FOR sti ← FIRST[STIndex], sti+SIZE[STRecord] UNTIL sti=stLimit DO
    WITH s:stb+sti SELECT FROM
      external =>
        WITH p:s SELECT FROM
          file =>
            IF p.fti = FTNull THEN
              BEGIN fti ← AddFile[s.hti]; p.fti ← fti END;
          ENDCASE;
      ENDCASE;
    ENDLOOP;
  fileArray ← DESCRIPTOR[SystemDefs.AllocateSegment[maxFiles],maxFiles];
  MiscDefs.SetBlock[BASE[fileArray],NIL,LENGTH[fileArray]];
  LookupFileTable[];
  TableDefs.DropNotify[Notifier];
  RETURN
  END;

EraseFileTable: PUBLIC PROCEDURE =
  BEGIN OPEN SegmentDefs;
  i: CARDINAL;
  f: FileHandle;
  FOR i IN [0..LENGTH[fileArray]) DO
```

```
    IF (f←fileArray[i]) # NIL THEN
      BEGIN UnlockFile[f]; IF f.segcount = 0 THEN ReleaseFile[f] END;
    ENDLOOP;
  SystemDefs.FreeSegment[BASE[fileArray]];
  END;

END...
```