

-- Directory.Mesa Edited by Sandman on August 23, 1977 9:40 PM

DIRECTORY

```

AltoDefs: FROM "altodefs",
BFSDefs: FROM "bfsdefs",
DirectoryDefs: FROM "directorydefs",
SegmentDefs: FROM "segmentdefs",
StringDefs: FROM "stringdefs",
StreamDefs: FROM "streamdefs",
AltoFileDefs: FROM "altofiledefs";

```

DEFINITIONS FROM SegmentDefs, StringDefs, AltoFileDefs, StreamDefs;

Directory: PROGRAM

```

IMPORTS BFSDefs, SegmentDefs, StreamDefs, StringDefs EXPORTS DirectoryDefs = BEGIN

```

```

FPptr: TYPE = POINTER TO FP;
DVptr: TYPE = POINTER TO DV;
HDptr: TYPE = POINTER TO HD;

```

```

BadFilename: PUBLIC SIGNAL [name:STRING] = CODE;
BadDirectory: PUBLIC SIGNAL [name:STRING] = CODE;

```

```

EnumerateDirectory: PUBLIC PROCEDURE [
  proc:PROCEDURE [POINTER TO FP, STRING] RETURNS [BOOLEAN]] =
  BEGIN
  PassItOn: PROCEDURE [i:StreamIndex, dv:DVptr, s:STRING] RETURNS [BOOLEAN] =
    BEGIN fp: FP;
      IF dv.type = DEfile THEN
        BEGIN
          BFSDefs.MakeFP[@fp,@dv.fp];
          RETURN[proc[@fp,s]]
        END;
      RETURN[FALSE]
    END;
  dir: StreamHandle ← CreateWordStream[SysDir,Read];
  [] ← EnumerateEntries[dir,PassItOn];
  dir.destroy[dir];
  RETURN
  END;

```

```

DirectoryLookup: PUBLIC PROCEDURE [fp:FPptr, name:STRING, create:BOOLEAN]
  RETURNS [old:BOOLEAN] =
  BEGIN
  dir: StreamHandle; access: AccessOptions; hd: HD;
  fn: STRING ← [FilenameChars]; ExpandFilename[name,fn];
  access ← IF ~create THEN Read ELSE Read+Write+Append;
  dir ← CreateWordStream[SysDir,access];
  old ← FindName[dir,fp,fn,@hd].found;
  IF ~old AND create THEN
    BEGIN
    -- should be @dir.file.fp
    BFSDefs.CreateFile[fn,fp,@SysDirFP];
    MakeEntry[dir,fp,fn,@hd];
    END;
  dir.destroy[dir];
  RETURN
  END;

```

```

DirectoryLookupFP: PUBLIC PROCEDURE [fp:FPptr, name:STRING]
  RETURNS [old:BOOLEAN] =
  BEGIN
  dir: StreamHandle ← CreateWordStream[SysDir,Read];
  old ← FindFP[dir,fp,name].found;
  dir.destroy[dir];
  RETURN
  END;

```

```

DirectoryPurge: PUBLIC PROCEDURE [fp:FPptr, name:STRING]
  RETURNS [found:BOOLEAN] =
  BEGIN
  dir: StreamHandle; index: StreamIndex;
  fn: STRING ← [FilenameChars]; ExpandFilename[name,fn];
  dir ← CreateWordStream[SysDir,Read+Write];
  [found,index] ← FindName[dir,fp,fn,NIL];
  IF found THEN DeleteEntry[dir,index];

```

```

dir.destroy[dir];
RETURN
END;

DirectoryPurgeFP: PUBLIC PROCEDURE [fp:FPptr]
RETURNS [found:BOOLEAN] =
BEGIN
dir: StreamHandle; index: StreamIndex;
dir ← CreateWordStream[SysDir,Read+Write];
[found,index] ← FindFP[dir,fp,NIL];
IF found THEN DeleteEntry[dir,index];
dir.destroy[dir];
RETURN
END;

-- Support Routines

DEugly: INTEGER = DEfile+1; -- for malformed DEfile entries.

BcplWords: INTEGER = 20; -- WordsForBcplString[FilenameChars];

HD: TYPE = RECORD [size, needed: INTEGER, index: StreamIndex];

EnumerateEntries: PUBLIC PROCEDURE [dir:StreamHandle,
proc:PROCEDURE [StreamIndex, DVptr, STRING] RETURNS [BOOLEAN]] RETURNS [StreamIndex] =
BEGIN
dv: DV; length: CARDINAL;
name: STRING ← [FilenameChars];
index: StreamIndex ← GetIndex[dir];
dn: ARRAY [0..BcplWords) OF UNSPECIFIED;
bcpl: POINTER TO bcplSTRING ← @dn[0];
UNTIL dir.endof[dir] DO
[] ← ReadBlock[dir,@dv,1];
IF (length ← dv.length)=0 THEN ERROR BadDirectory[name];
IF dv.type = DEfile THEN
IF length IN (SIZE[DV]..SIZE[DV]+LENGTH[dn]) THEN
BEGIN
[] ← ReadBlock[dir,@dv+1,SIZE[DV]-1];
[] ← ReadBlock[dir,bcpl,length-SIZE[DV]];
BcplToMesaString[bcpl,name];
END
ELSE dv.type ← DEugly;
IF proc[index,@dv,name] THEN EXIT;
index.byte ← index.byte+length*AltoDefs.BytesPerWord;
SetIndex[dir,index ← NormalizeIndex[index]];
ENDLOOP;
RETURN[index]
END;

sinkHD: HD;

FindName: PROCEDURE [dir:StreamHandle, fp:FPptr, name:STRING, hd:HDptr]
RETURNS [found:BOOLEAN, index:StreamIndex] =
BEGIN
MatchName: PROCEDURE [i:StreamIndex, dv:DVptr, s:STRING]RETURNS [BOOLEAN] =
BEGIN
IF hd.size=0 THEN hd.index ← i;
SELECT dv.type FROM
DEfree =>
IF hd.size < hd.needed
THEN hd.size ← hd.size+dv.length;
DEfile =>
IF found ← [equivalentString[name,s]
THEN BFSDefs.MakeFP[fp,@dv.fp];
ENDCASE;
-- SIGNAL BadDirectory[s];
IF dv.type # DEfree
AND hd.size < hd.needed
THEN hd.size ← 0;
RETURN[found]
END;
IF hd = NIL THEN hd ← @sinkHD;
hdr ← HD[0,SIZE[DV]+WordsForBcplString[name.length].];
found ← FALSE; index ← [enumerateEntries[dir,MatchName];
IF hd.size=0 THEN hd.index ← index;

```

```

RETURN
END;

FindFP: PROCEDURE [dir:StreamHandle, fp:FPptr, name:STRING]
RETURNS [found:BOOLEAN, index:StreamIndex] =
BEGIN
MatchFP: PROCEDURE [i:StreamIndex, dv:DVptr, s:STRING] RETURNS [BOOLEAN] =
BEGIN f: FP;
SELECT dv.type FROM
DEfree => NULL;
DEfile =>
BEGIN BFSDefs.MakeFP[@f,@dv.fp];
IF (found ← f=fp↑)
AND name # NIL THEN
BEGIN name.length ← 0;
AppendString[name,s];
END;
END;
ENDCASE; -- SIGNAL BadDirectory[s];
RETURN[found]
END;
found ← FALSE;
index ← EnumerateEntries[dir,MatchFP];
RETURN
END;

MakeEntry: PROCEDURE [dir:StreamHandle, fp:FPptr, name:STRING, hd:HDptr] =
BEGIN leftover: CARDINAL;
dv: DV ← DV[DEfile,hd.needed,];
dn: ARRAY [0..BcplWords) OF UNSPECIFIED;
bcpl: POINTER TO bcplSTRING ← @dn[0];
IF name.length > FilenameChars
THEN [ERROR BadFilename[name];
BFSDefs.MakeCFP[@dv.fp,fp];
MesaToBcplString[name,bcpl];
SetIndex[dir,hd.index];
[] ← WriteBlock[dir,@dv,SIZE[DV]];
[] ← WriteBlock[dir,bcpl,hd.needed-SIZE[DV]];
IF (leftover ← hd.size-hd.needed)>0 THEN
BEGIN
dv ← DV[DEfree,leftover,];
[] ← WriteBlock[dir,@dv,1];
END;
RETURN
END;

DeleteEntry: PROCEDURE [dir:StreamHandle, index:StreamIndex] =
BEGIN dv:DV;
SetIndex[dir,index];
[] ← ReadBlock[dir,@dv,1];
dv.type ← DEfree;
SetIndex[dir,index];
[] ← WriteBlock[dir,@dv,1];
RETURN
END;

-- Filename Parsing (such as it is)

ExpandFilename: PUBLIC PROCEDURE [name,filename:STRING] =
BEGIN filename.length ← 0;
AppendString[filename,name];
IF filename[filename.length-1] # '.'
THEN AppendChar[filename,'];
RETURN
END;

-- Main Body

SysDir: FileHandle ← NIL;
SysDirFP: FP ← FP[SN[1,0,0,0,DirSN],DirDA];

LockFile[SysDir ← InsertFile[@SysDirFP,Read+Write+Append]];

END.

```

-- stuff yet to be done:

SetCurrentDir, CloseCurrentDir, SetWorkingDir, ParseFilename.

VersionsKept, StripVersion, AppendVersion, SearchForVersion.