```
-- MakeImageUtilities.Mesa  Edited by Sandman on May 17, 1978  8:54 AM

DIRECTORY
  AltoDefs: FROM "altodefs" USING [PageSize],
  AltoFileDefs: FROM "altofiledefs" USING [DirSN, eofDA, FA, FP, SN],
  BcdMergeDefs: FROM "bcdmergedefs" USING [
    FinalizeMerge, InitializeMerge, MergeBcd, MergedBcdSize, MergeModule,
    WriteMergedBcd],
  ControlDefs: FROM "controldefs" USING [
    GFT, GFTIndex, GFTItem, NullGlobalFrame],
  DirectoryDefs: FROM "directorydefs" USING [EnumerateDirectory],
  FrameDefs: FROM "framedefs" USING [
    EnumerateGlobalFrames, MakeCodeResident, SwapInCode, SwapOutCode],
  ImageDefs: FROM "imagedefs" USING [ImageHeader, MapItem],
  LoaderBcdUtilDefs: FROM "loaderbcdutildefs" USING [
    BcdBase, ReleaseBcdSeg, SetUpBcd],
  LoadStateDefs: FROM "loadstatedefs" USING [
    BcdAddress, BcdSegFromLoadState, ConfigGFI, ConfigIndex, ConfigNull,
    EnumerateLoadStateBcds, FileSegmentHandle, InitializeRelocation,
    MapRealToConfig, ReleaseRelocation, Relocation],
  MiscDefs: FROM "miscdefs" USING [SetBlock],
  MIUtilityDefs: FROM "miutilitydefs" USING [
    CFA, ConfigIndex, DataSegmentHandle, FileHandle, FileRequest,
    FileSegmentHandle, FP, GlobalFrameHandle, LoadStateGFT, MoveWords,
    PageNumber, ProcDesc, SpaceHeader, vDA],
  SDDefs: FROM "sddefs" USING [SD, sGFTLength],
  SegmentDefs: FROM "segmentdefs" USING [
    AddressFromPage, DataSegmentAddress, DefaultBase, DefaultVersion,
    DeleteDataSegment, EnumerateFileSegments, FileHandle, FileHint,
    FileSegmentHandle, InsertFile, JumpToPage, NewDataSegment, NewFile,
    NewFileSegment, Read, SwapOut, Unlock, Write],
  StreamDefs: FROM "streamdefs" USING [
    DiskHandle, GetIndex, SetIndex, StreamIndex],
  StringDefs: FROM "stringdefs" USING [
    EquivalentString, EquivalentSubStrings, SubStringDescriptor],
  SystemDefs: FROM "systemdefs" USING [AllocatePages, FreePages];

DEFINITIONS FROM ImageDefs, MIUtilityDefs;

MakeImageUtilities: PROGRAM
  IMPORTS DirectoryDefs, FrameDefs, LoadStateDefs, MiscDefs, SegmentDefs,
    StreamDefs, StringDefs, SystemDefs, BcdMergeDefs, MIUtilityDefs,
    LoaderBcdUtilDefs
  EXPORTS ImageDefs, MIUtilityDefs
  SHARES SegmentDefs, ControlDefs, ImageDefs =

PUBLIC BEGIN


-- file requests

  RequestHead: POINTER TO FileRequest;

  InitFileRequest: PROCEDURE = BEGIN  RequestHead ← NIL; END;

  AddFileRequest: PROCEDURE [r: POINTER TO FileRequest] =
    BEGIN
    r.link ← RequestHead;
    RequestHead ← r;
    END;

  DropFileRequest: PROCEDURE [f: FileHandle] =
    BEGIN
    r: POINTER TO FileRequest;
    prev: POINTER TO FileRequest ← NIL;
    FOR r ← RequestHead, r.link UNTIL r = NIL DO
      BEGIN
      IF r.file = f THEN
        IF prev = NIL THEN RequestHead ← r.link
        ELSE prev.link ← r.link;
        EXIT;
        END;
      prev ← r;
      ENDLOOP;
    END;
```

```
  ProcessFileRequests: PROCEDURE =
    BEGIN OPEN AltoFileDefs;
    checkone: PROCEDURE [fp: POINTER TO FP, dname: STRING] RETURNS [BOOLEAN] =
      BEGIN
      ss: StringDefs.SubStringDescriptor ← [dname,0,dname.length];
      r: POINTER TO FileRequest;
      prev: POINTER TO FileRequest ← NIL;
      FOR r ← RequestHead, r.link UNTIL r = NIL DO
        IF (WITH r SELECT FROM
              long => StringDefs.EquivalentSubStrings[@ss,@name],
              short => StringDefs.EquivalentString[dname,name],
              ENDCASE => FALSE) THEN
          BEGIN
          IF r.file = NIL THEN r.file ← SegmentDefs.InsertFile[fp,r.access]
          ELSE r.file.fp ← fp↑;
          IF prev = NIL THEN RequestHead ← r.link
          ELSE prev.link ← r.link;
          END
        ELSE prev ← r;
        ENDLOOP;
      RETURN[RequestHead = NIL]
      END;

    DirectoryDefs.EnumerateDirectory[checkone];
    END;

-- bcd file names

GetBcdFileNames: PROCEDURE  [nbcds: ConfigIndex]
RETURNS [names: DESCRIPTOR FOR ARRAY OF STRING] =
  BEGIN
  nfound: ConfigIndex ← 0;
  GetNames: PROCEDURE [fp: POINTER TO FP, s: STRING] RETURNS[BOOLEAN] =
    BEGIN
    FindBcd: PROCEDURE [config: ConfigIndex, bcd: LoadStateDefs.BcdAddress] RETURNS [BOOLEAN] =
      BEGIN
      IF fp↑ = bcd.fp THEN
        BEGIN
        names[config] ← GetString[s];
        nfound ← nfound + 1;
        RETURN[TRUE];
        END;
      RETURN[FALSE];
      END;
    [] ← LoadStateDefs.EnumerateLoadStateBcds[recentfirst, FindBcd];
    RETURN[nfound = nbcds];
    END;
  names ← DESCRIPTOR[GetSpace[nbcds], nbcds];
  MiscDefs.SetBlock[BASE[names], GetString["(anon)"], nbcds];
  DirectoryDefs.EnumerateDirectory[GetNames];
  RETURN[names];
  END;

-- space allocation

SpaceList: POINTER TO SpaceHeader ← NIL;
SpacePointer: POINTER TO SpaceHeader;
SpaceLeft: CARDINAL;

InitSpace: PROCEDURE = BEGIN SpaceLeft ← 0; END;

GetSpace: PROCEDURE [n: CARDINAL] RETURNS [p: POINTER] =
  BEGIN
  newseg: DataSegmentHandle;
  IF n > SpaceLeft THEN
    BEGIN
    newseg ← SegmentDefs.NewDataSegment[SegmentDefs.DefaultBase,1];
    SpacePointer ← SegmentDefs.DataSegmentAddress[newseg];
    SpacePointer.link ← SpaceList;
    SpacePointer.segment ← newseg;
    SpaceList ← SpacePointer;
    SpacePointer ← SpacePointer + SIZE[SpaceHeader];
    SpaceLeft ← AltoDefs.PageSize - SIZE[SpaceHeader];
    END;
  p ← SpacePointer;
  SpacePointer ← SpacePointer + n;
```

```
  SpaceLeft ← SpaceLeft - n;
  END;

GetString: PROCEDURE [oldstring: STRING] RETURNS [newstring: STRING] =
  BEGIN
  i, length: CARDINAL;
  string: TYPE = POINTER TO MACHINE DEPENDENT RECORD[length,maxlength: CARDINAL];
  length ← oldstring.length;
  newstring ← GetSpace[(length+5)/2];
  newstring.length ← length;
  LOOPHOLE[newstring, string].maxlength ← length;
  FOR i IN [0..length) DO newstring[i] ← oldstring[i]; ENDLOOP;
  RETURN;
  END;

FreeAllSpace: PROCEDURE =
  BEGIN
  next: POINTER TO SpaceHeader;
  UNTIL SpaceList = NIL DO
    next ← SpaceList.link;
    SegmentDefs.DeleteDataSegment[SpaceList.segment];
    SpaceList ← next;
    ENDLOOP;
  END;

-- image file management

LockCodeSegment: PROCEDURE [p: ProcDesc] =
  BEGIN OPEN FrameDefs;
  frame: GlobalFrameHandle ← ControlDefs.GFT[p.gfi].frame;
  MakeCodeResident[frame];
  SwapInCode[frame];
  SegmentDefs.Unlock[frame.codesegment];
  END;

UnlockCodeSegment: PROCEDURE [p: ProcDesc] =
  BEGIN
  SegmentDefs.Unlock[ControlDefs.GFT[p.gfi].frame.codesegment];
  END;

KDSegment: PROCEDURE RETURNS [FileSegmentHandle] =
  BEGIN OPEN SegmentDefs;
  DiskKDFile: FileHandle = NewFile["DiskDescriptor", Read, DefaultVersion];
  FindKD: PROCEDURE [s: FileSegmentHandle] RETURNS [BOOLEAN] =
    BEGIN
    RETURN[s.file = DiskKDFile];
    END;
  RETURN[SegmentDefs.EnumerateFileSegments[FindKD]];
  END;

DAofPage: PROCEDURE [file: FileHandle, page: PageNumber] RETURNS [next: vDA] =
  BEGIN
  cfa: CFA;
  buf: POINTER = SystemDefs.AllocatePages[1];
  cfa.fp ← file.fp;
  cfa.fa ← AltoFileDefs.FA[file.fp.leaderDA,0,0];
  next ← SegmentDefs.JumpToPage[@cfa,page-1,buf].next;
  SystemDefs.FreePages[buf];
  RETURN
  END;

FillInCAs: PROCEDURE [
  Image: POINTER TO ImageHeader, mapindex: CARDINAL, ca: POINTER] =
  BEGIN
  i: CARDINAL;
  map: POINTER TO ARRAY [0..0) OF normal MapItem = LOOPHOLE[@Image.map];
  addr: POINTER;
  FOR i IN [0..mapindex) DO
    addr ← SegmentDefs.AddressFromPage[map[i].page];
    THROUGH [0..map[i].count) DO
      ca↑ ← addr;
      ca ← ca + 1;
      addr ← addr + AltoDefs.PageSize;
      ENDLOOP;
    ENDLOOP;
  END;
```

```
SwapOutUnlockedCode: PROCEDURE [f: GlobalFrameHandle] RETURNS [BOOLEAN] =
  BEGIN
  cseg: FileSegmentHandle ← f.codesegment;
  IF cseg.swappedin AND cseg.lock = 0 THEN FrameDefs.SwapOutCode[f];
  RETURN[FALSE]
  END;

SwapOutUnlocked: PROCEDURE [s: FileSegmentHandle] RETURNS [BOOLEAN] =
  BEGIN
  IF s.lock = 0 THEN SegmentDefs.SwapOut[s];
  RETURN[FALSE];
  END;

BashHint: PROCEDURE [s: FileSegmentHandle] RETURNS [BOOLEAN] =
  BEGIN
  WITH s SELECT FROM
    disk => hint ← SegmentDefs.FileHint[da:AltoFileDefs.eofDA, page:0];
    ENDCASE;
  RETURN[FALSE];
  END;

BashFile: PROCEDURE [f: FileHandle] RETURNS [BOOLEAN] =
  BEGIN OPEN AltoFileDefs;
  f.open ← f.lengthvalid ← FALSE;
  IF f.fp.serial # SN[1,0,0,0,DirSN] THEN
    f.fp ← FP[SN[1,0,1,17777B,177777B],eofDA];
  RETURN[FALSE];
  END;

PatchUpGFT: PROCEDURE =
  BEGIN OPEN ControlDefs;
  i: GFTIndex;
  gft: POINTER TO ARRAY [0..0) OF GFTItem ← GFT;
  FOR i IN [1..SDDefs.SD[SDDefs.sGFTLength]) DO
    IF gft[i] = [frame: NullGlobalFrame, epbase: 177777B] THEN
      gft[i] ← [frame: NullGlobalFrame, epbase: 0];
    ENDLOOP;
  RETURN
  END;

InitLoadStateGFT: PROCEDURE [initgft: LoadStateGFT, merge: BOOLEAN, nbcds: ConfigIndex] =
  BEGIN OPEN ControlDefs, LoadStateDefs;
  rgfi, cgfi: GFTIndex ← 0;
  i: ConfigIndex;
  gft: POINTER TO ARRAY [0..0) OF GFTItem ← GFT;
  gftLength: CARDINAL = SDDefs.SD[SDDefs.sGFTLength];
  MiscDefs.SetBlock[
    p: BASE[initgft], v: ConfigGFI[config: ConfigNull, gfi: 0], 1: gftLength];
  IF merge THEN
    FOR rgfi IN [1..gftLength) DO
      IF gft[rgfi].frame # NullGlobalFrame THEN initgft[rgfi] ←
        [config: 0, gfi: (cgfi←cgfi+1)];
      ENDLOOP
  ELSE
    FOR i IN [0..nbcds) DO
      cgfi ← 0;
      FOR rgfi IN [1..gftLength) DO
        IF gft[rgfi].frame # NullGlobalFrame AND
          LoadStateDefs.MapRealToConfig[rgfi].config = i
          THEN initgft[rgfi] ← [config: i, gfi: (cgfi←cgfi+1)];
        ENDLOOP;
      ENDLOOP;
  END;

NumberGFIInConfig: PROCEDURE [initgft: LoadStateGFT, con: ConfigIndex]
RETURNS [ngfi: ControlDefs.GFTIndex] =
  BEGIN
  i: ControlDefs.GFTIndex;
  ngfi ← 0;
  FOR i IN [0..LENGTH[initgft]) DO
    IF initgft[i].config = con THEN ngfi ← ngfi + 1;
    ENDLOOP;
  RETURN
  END;
```

```
-- Bcd Merging Management

TableSize: CARDINAL = 15*AltoDefs.PageSize;

MergeAllBcds: PROCEDURE [initialgft: LoadStateGFT, code: BOOLEAN,
  names: DESCRIPTOR FOR ARRAY OF STRING] =
  BEGIN OPEN LoadStateDefs, BcdMergeDefs;
  MergeLoadedBcds: PROCEDURE [config: ConfigIndex, addr: BcdAddress] RETURNS [BOOLEAN] =
    BEGIN OPEN LoaderBcdUtilDefs, LoadStateDefs;
    rel: Relocation ← InitializeRelocation[config];
    bcdseg: FileSegmentHandle ← BcdSegFromLoadState[config];
    bcd: BcdBase ← SetUpBcd[bcdseg];
    MergeBcd[bcd, rel, 0, initialgft, code, names[config]];
    ReleaseBcdSeg[bcdseg];
    ReleaseRelocation[rel];
    RETURN [FALSE];
    END;

  MergeCopiedFrames: PROCEDURE [frame: GlobalFrameHandle] RETURNS [BOOLEAN] =
    BEGIN
    copied: GlobalFrameHandle;
    config: ConfigIndex;
    ModuleCopiedFrom: PROCEDURE [f: GlobalFrameHandle] RETURNS [BOOLEAN] =
      BEGIN
      RETURN [f # frame AND f.codesegment = frame.codesegment AND
        (config ← MapRealToConfig[f.gfi].config) # ConfigNull];
      END;
    IF MapRealToConfig[frame.gfi].config # ConfigNull THEN RETURN [FALSE];
    IF (copied←FrameDefs.EnumerateGlobalFrames[ModuleCopiedFrom]) #
      ControlDefs.NullGlobalFrame THEN
      BEGIN
      MergeModule[frame, copied, initialgft];
      RETURN [FALSE];
      END;
    RETURN [FALSE];
    END;

  InitializeMerge[TableSize, NumberGFIInConfig[initialgft, 0]];
  [] ← EnumerateLoadStateBcds[recentlast, MergeLoadedBcds];
  [] ← FrameDefs.EnumerateGlobalFrames[MergeCopiedFrames];
  [] ← MergedBcdSize[];
  WriteMergedBcd[MoveWords];
  FinalizeMerge[];
  END;

MergeABcd: PROCEDURE [config: ConfigIndex, initgft: LoadStateGFT, code: BOOLEAN,
  names: DESCRIPTOR FOR ARRAY OF STRING] =
  BEGIN OPEN BcdMergeDefs, LoaderBcdUtilDefs, LoadStateDefs;
  rel: Relocation ← InitializeRelocation[config];
  bcdseg: FileSegmentHandle ← BcdSegFromLoadState[config];
  bcd: BcdBase ← SetUpBcd[bcdseg];
  InitializeMerge[bcdseg.pages+1, NumberGFIInConfig[initgft, config]];
  MergeBcd[bcd, rel, config, initgft, code, names[config]];
  ReleaseBcdSeg[bcdseg];
  ReleaseRelocation[rel];
  [] ← MergedBcdSize[];
  WriteMergedBcd[MoveWords];
  FinalizeMerge[];
  END;

NewBcdSegmentFromStream: PROCEDURE [stream: StreamDefs.DiskHandle, page: PageNumber]
  RETURNS [newpage: PageNumber, seg: FileSegmentHandle] =
  BEGIN OPEN SegmentDefs;
  index: StreamDefs.StreamIndex;
  index ← StreamDefs.GetIndex[stream];
  IF index.byte # 0 THEN
    BEGIN
    index.byte ← 0;
    index.page ← index.page + 1;
    StreamDefs.SetIndex[stream, index];
    END;
  seg ← NewFileSegment[stream.file, page+1, index.page-page, Read+Write];
  newpage ← index.page;
  RETURN
  END;
```

```
END..
```