```
-- LoadState.mesa
-- Last Modified by Sandman,  May 12, 1978  2:34 PM

DIRECTORY
  AltoFileDefs: FROM "altofiledefs" USING [eofDA],
  InlineDefs: FROM "inlinedefs" USING [COPY],
  LoadStateDefs: FROM "loadstatedefs" USING [
    BcdAddress, BcdArrayLength, ConfigGFI, ConfigIndex, ConfigNull,
    EnumerationDirection, FileSegmentHandle, GFTIndex, LoadState,
    LoadStateGFT, Relocation],
  MiscDefs: FROM "miscdefs" USING [SetBlock, Zero],
  NucleusDefs: FROM "nucleusdefs",
  SDDefs: FROM "sddefs" USING [SD, sGFTLength],
  SegmentDefs: FROM "segmentdefs" USING [
    FileHandle, FileHint, FileSegmentAddress, GetFileFP, InsertFile,
    NewFileSegment, Read, SwapIn, SwapOut, Unlock],
  SystemDefs: FROM "systemdefs" USING [AllocateHeapNode, FreeHeapNode];

DEFINITIONS FROM LoadStateDefs;

LoadState: PROGRAM [state, initstate, bcdseg: FileSegmentHandle]
  IMPORTS MiscDefs, SegmentDefs, SystemDefs
  EXPORTS LoadStateDefs, NucleusDefs = PUBLIC

  BEGIN

  loadstate: LoadState;
  gft: LoadStateGFT;
  nbcds: ConfigIndex;

  LoadStateFull: ERROR = CODE;

  InputLoadState: PROCEDURE RETURNS [ConfigIndex] =
    BEGIN OPEN SegmentDefs;
    i: GFTIndex;
    SwapIn[state];
    loadstate ← FileSegmentAddress[state];
    gft ← DESCRIPTOR[@loadstate.gft, SDDefs.SD[SDDefs.sGFTLength]];
    nbcds ← 0;
    FOR i IN [0..LENGTH[gft]) DO
      IF gft[i].config # ConfigNull THEN nbcds ← MAX[nbcds, gft[i].config];
      ENDLOOP;
    nbcds ← nbcds + 1;
    RETURN[nbcds]
    END;

  UpdateLoadState: PROCEDURE [
   bcd: ConfigIndex, bcdseg: FileSegmentHandle, unresolved, exports: BOOLEAN] =
    BEGIN OPEN SegmentDefs;
    IF bcd >= LAST[ConfigIndex] THEN ERROR LoadStateFull;
    loadstate.bcds[bcd] ←
      [fp:, da:, base: bcdseg.base, unresolved: unresolved,
      fill:, exports: exports, pages: bcdseg.pages];
    loadstate.bcds[bcd].da ← WITH s: bcdseg SELECT FROM
      disk => s.hint.da,
      ENDCASE => AltoFileDefs.eofDA;
    GetFileFP[bcdseg.file, @loadstate.bcds[bcd].fp];
    nbcds ← nbcds + 1;
    END;

RemoveConfig: PUBLIC PROCEDURE [rel: Relocation, config: ConfigIndex] =
  BEGIN
  i: CARDINAL;
  FOR i IN [1..LENGTH[gft]) DO
    IF gft[i].config > config AND gft[i].config # ConfigNull THEN
      gft[i].config ← gft[i].config - 1;
    ENDLOOP;
  FOR i IN [1..LENGTH[rel]) DO
    gft[rel[i]] ← ConfigGFI[ConfigNull, 0]; ENDLOOP;
  FOR i IN [config..nbcds) DO
    loadstate.bcds[i] ← loadstate.bcds[i+1];
    ENDLOOP;
  END;

  ReleaseLoadState: PROCEDURE =
    BEGIN OPEN SegmentDefs;
    IF ~state.swappedin THEN RETURN;
    Unlock[state];
    IF state.lock = 0 THEN
      BEGIN
      SwapOut[state];
      loadstate ← NIL;
      nbcds ← 0;
      END;
    END;

  EnterGfi: PROCEDURE [cgfi, rgfi: GFTIndex, config: ConfigIndex] =
    BEGIN
    gft[rgfi] ← [config: config, gfi: cgfi];
    END;

  MapConfigToReal: PROCEDURE [cgfi: GFTIndex, config: ConfigIndex] RETURNS [rgfi: GFTIndex] =
    BEGIN
    IF cgfi = 0 THEN RETURN[0];
    FOR rgfi IN [0..LENGTH[gft]) DO
      IF gft[rgfi] = [config, cgfi] THEN RETURN [rgfi];
      ENDLOOP;
    RETURN[0];
    END;

  MapRealToConfig: PROCEDURE [rgfi: GFTIndex] RETURNS [cgfi: GFTIndex, config: ConfigIndex] =
    BEGIN
    RETURN[gft[rgfi].gfi, gft[rgfi].config];
    END;

  InitializeRelocation: PROCEDURE [config: ConfigIndex] RETURNS [rel: Relocation] =
    BEGIN
    max: CARDINAL ← 0;
    i: GFTIndex;
    FOR i IN [0..LENGTH[gft]) DO
      IF gft[i].config = config THEN max ← MAX[max, gft[i].gfi];
      ENDLOOP;
    rel ← DESCRIPTOR[SystemDefs.AllocateHeapNode[max+1], max+1];
    MiscDefs.Zero[BASE[rel], max+1];
    FOR i IN [0..LENGTH[gft]) DO
      IF gft[i].config = config THEN rel[gft[i].gfi] ← i;
      ENDLOOP;
    END;

  ReleaseRelocation: PROCEDURE [rel: Relocation] =
    BEGIN
    SystemDefs.FreeHeapNode[BASE[rel]];
    END;

  BcdSegFromLoadState: PROCEDURE [bcd: ConfigIndex] RETURNS [seg: FileSegmentHandle] =
    BEGIN OPEN SegmentDefs, b: loadstate.bcds[bcd];
    bcdfile: FileHandle ← InsertFile[@b.fp, Read];
    seg ← NewFileSegment[bcdfile, b.base, b.pages, Read];
    IF b.da # AltoFileDefs.eofDA THEN
      WITH s: seg SELECT FROM
        disk => s.hint ← SegmentDefs.FileHint[b.da, b.base];
        ENDCASE;
    RETURN
    END;

  UpdateLoadStateDA: PROCEDURE [bcdseg: FileSegmentHandle] =
    BEGIN OPEN SegmentDefs;
    FindSeg: PROCEDURE [c: ConfigIndex, b: BcdAddress] RETURNS [BOOLEAN] =
      BEGIN
      IF b.base = bcdseg.base AND b.pages = bcdseg.pages
        AND b.fp = bcdseg.file.fp THEN
        BEGIN
        WITH s: bcdseg SELECT FROM
          disk => IF s.hint.da # AltoFileDefs.eofDA THEN b.da ← s.hint.da;
          ENDCASE;
        RETURN[TRUE];
        END;
      RETURN[FALSE];
      END;
    IF loadstate = NIL THEN RETURN;  -- loadstate not in
    [] ← EnumerateLoadStateBcds[recentfirst, FindSeg];
    RETURN
    END;

  EnumerateLoadStateGFT: PROCEDURE [
   proc: PROCEDURE [GFTIndex, GFTIndex, ConfigIndex] RETURNS [BOOLEAN]]
    RETURNS [GFTIndex] =
    BEGIN
    i: GFTIndex;
    FOR i IN [0..LENGTH[gft]) DO
      IF proc[i, gft[i].gfi, gft[i].config] THEN RETURN[i];
      ENDLOOP;
    RETURN[0]
    END;

  EnumerateLoadStateBcds: PROCEDURE [dir: EnumerationDirection,
   proc: PROCEDURE [ConfigIndex, BcdAddress] RETURNS [BOOLEAN]]
    RETURNS [config: ConfigIndex, bcd: BcdAddress] =
    BEGIN
    i: CARDINAL;
    SELECT dir FROM
      recentfirst =>
        FOR i DECREASING IN [0..nbcds) DO
          IF proc[i, @loadstate.bcds[i]] THEN RETURN[i, @loadstate.bcds[i]];
          ENDLOOP;
      recentlast =>
        FOR i IN [0..nbcds) DO
          IF proc[i, @loadstate.bcds[i]] THEN RETURN[i, @loadstate.bcds[i]];
          ENDLOOP;
      ENDCASE;
    RETURN[ConfigNull, NIL]
    END;

  BcdHasUnresolvedImports: PROCEDURE [bcd: ConfigIndex] RETURNS [BOOLEAN] =
    BEGIN
    RETURN[loadstate.bcds[bcd].unresolved];
    END;

  SetUnresolvedImports: PROCEDURE [bcd: ConfigIndex, unresolved: BOOLEAN] =
    BEGIN
    loadstate.bcds[bcd].unresolved ← unresolved;
    END;

  BcdHasExports: PROCEDURE [bcd: ConfigIndex] RETURNS [BOOLEAN] =
    BEGIN
    RETURN[loadstate.bcds[bcd].exports];
    END;

  SetLoadState: PROCEDURE [stateseg: FileSegmentHandle] =
    BEGIN
    state ← stateseg;
    END;

  GetLoadState: PROCEDURE RETURNS [FileSegmentHandle] =
    BEGIN
    RETURN[state];
    END;

  GetInitialLoadState: PROCEDURE RETURNS [FileSegmentHandle] =
    BEGIN
    RETURN[initstate];
    END;

  ResetLoadState: PROCEDURE [initialGFT: LoadStateGFT] =
    BEGIN
    MiscDefs.Zero[loadstate, BcdArrayLength];
    MiscDefs.SetBlock[
      p: @loadstate.gft,
      v: ConfigGFI[config: ConfigNull, gfi: 0],
      l: LENGTH[gft]];
    InlineDefs.COPY[
      from: BASE[initialGFT], to: BASE[gft], nwords: LENGTH[gft]];
    nbcds ← 0;
    END;


  InitLoadStateObject: PROCEDURE =
    BEGIN OPEN SegmentDefs;
    initloadstate: LoadState ← FileSegmentAddress[initstate];
    loadstate ← FileSegmentAddress[state];
    gft ← DESCRIPTOR[@loadstate.gft, SDDefs.SD[SDDefs.sGFTLength]];
    InlineDefs.COPY[
      from: initloadstate, to: loadstate, nwords: LENGTH[gft]+BcdArrayLength];
    loadstate.bcds[0].fp ← bcdseg.file.fp;
    loadstate.bcds[0].base ← bcdseg.base;
    WITH s: bcdseg SELECT FROM
      disk => loadstate.bcds[0].da ← s.hint.da;
      ENDCASE;
    ReleaseLoadState[];
    Unlock[initstate];
    SwapOut[initstate];
    END;

-- Main Body

  InitLoadStateObject[];

END..
```