```
-- SegmentDefs.Mesa  Edited by Sandman on August 23, 1977  9:16 PM

DIRECTORY
  AltoDefs: FROM "altodefs",
  AltoFileDefs: FROM "altofiledefs";

DEFINITIONS FROM AltoDefs, AltoFileDefs;

SegmentDefs: DEFINITIONS = BEGIN

  FileHint: TYPE = AltoFileDefs.FH;
  FileIndex: TYPE = AltoFileDefs.FI;
  PageCount: TYPE = AltoDefs.PageCount;
  PageNumber: TYPE = AltoDefs.PageNumber;


  -- Basic Memory Addressing:

  PagePointer: PROCEDURE [a:POINTER] RETURNS [POINTER];
  PageFromAddress: PROCEDURE [a:POINTER] RETURNS [PageNumber];
  AddressFromPage: PROCEDURE [p:PageNumber] RETURNS [POINTER];

  PageFree: PROCEDURE [page: PageNumber] RETURNS [BOOLEAN];
  PagesFree: PROCEDURE [base: PageNumber, pages: PageCount] RETURNS [BOOLEAN];


  -- Primative System Objects:

  ObjectType: PRIVATE TYPE = {Free, Data, Segment, File};

  ObjectHandle: PRIVATE TYPE = POINTER TO Object;

  Object: TYPE = RECORD [
    inuse, busy: PRIVATE BOOLEAN,
    body: SELECT COMPUTED ObjectType FROM
      Free => [
        fill: PRIVATE [0..37777B],
        link: PRIVATE POINTER TO Free Object],
      Data => [
        pages: [1..MaxVMPage+1],
        VMpage: [0..MaxVMPage]],
      Segment => [
        swappedin: BOOLEAN,
        read, write: BOOLEAN,
        class: FileSegmentClass,
        lock: [0..MaxLocks],
        pages: [1..MaxVMPage+1],
        VMpage: [0..MaxVMPage],
        file: FileHandle,
        base: PageNumber,
        hint: FileHint],
      File => [
        open: BOOLEAN,
        lengthvalid: BOOLEAN,
        lengthchanged: BOOLEAN,
        read, write, append: BOOLEAN,
        lock: [0..MaxLocks],
        swapcount: [0..MaxRefs],
        segcount: [0..MaxSegs],
        fp: FP
        eof: FA],
      ENDCASE];

  MaxRefs: CARDINAL = 255;
  MaxLocks: CARDINAL = 255;
  MaxSegs: CARDINAL = 177777B;
```

```
  -- Segments:

  DefaultPages: PageCount = 0;
  DefaultBase: PageNumber = MaxFilePage;


  -- Data Segments:

  DataSegmentObject: TYPE = Data Object;
  DataSegmentHandle: TYPE = POINTER TO DataSegmentObject;

  VMNotFree: SIGNAL [base:PageNumber, pages:PageCount];

  NewDataSegment: PROCEDURE [base:PageNumber, pages:PageCount] RETURNS [DataSegmentHandle];
  NewFrameSegment: PROCEDURE [pages: PageCount] RETURNS [DataSegmentHandle];
  DeleteDataSegment: PROCEDURE [seg:DataSegmentHandle];

  VMtoDataSegment: PROCEDURE [a:POINTER] RETURNS [DataSegmentHandle];
  DataSegmentAddress: PROCEDURE [seg:DataSegmentHandle] RETURNS [POINTER];

  EnumerateDataSegments: PROCEDURE [
    proc: PROCEDURE [DataSegmentHandle] RETURNS [BOOLEAN]]
    RETURNS [DataSegmentHandle];


  -- File Segments:

  FileSegmentObject: TYPE = Segment Object;
  FileSegmentHandle: TYPE = POINTER TO FileSegmentObject;

  FileSegmentClass: TYPE = {code, symbols, bcd, other};

  InvalidSegmentSize: SIGNAL [pages:PageCount];

  NewFileSegment: PROCEDURE [
    file:FileHandle, base:PageNumber, pages:PageCount, access:AccessOptions]
    RETURNS [FileSegmentHandle];
  MoveFileSegment: PROCEDURE [seg:FileSegmentHandle, base:PageNumber, pages:PageCount];
  MapFileSegment: PROCEDURE [seg: FileSegmentHandle, file: FileHandle, base: PageNumber];
  DeleteFileSegment: PROCEDURE [seg:FileSegmentHandle];

  VMtoFileSegment: PROCEDURE [a:POINTER] RETURNS [FileSegmentHandle];
  FileSegmentAddress: PROCEDURE [seg:FileSegmentHandle] RETURNS [POINTER];
  GetFileSegmentDA: PROCEDURE [seg: FileSegmentHandle] RETURNS [vDA];
  SetFileSegmentDA: PROCEDURE [seg: FileSegmentHandle, da: vDA];

  EnumerateFileSegments: PROCEDURE [
    proc: PROCEDURE [FileSegmentHandle] RETURNS [BOOLEAN]]
    RETURNS [FileSegmentHandle];


  -- File Segment Swapping:

  SwapError: SIGNAL [seg:FileSegmentHandle];
  SegmentFault: SIGNAL [seg:FileSegmentHandle, pages:PageCount];
  SwapIn, SwapUp, SwapOut, Unlock: PROCEDURE [seg:FileSegmentHandle];

  -- Initializing File and Data Segments

  CopyDataToFileSegment: PROCEDURE [dataseg: DataSegmentHandle, fileseg: FileSegmentHandle];
  CopyFileToDataSegment: PROCEDURE [fileseg: FileSegmentHandle, dataseg: DataSegmentHandle];
  ChangeDataToFileSegment: PROCEDURE [dataseg: DataSegmentHandle, fileseg: FileSegmentHandle];

  -- Swap Strategies and Swapping Procedures:

  InsufficientVM: SIGNAL [needed:PageCount];

  TryCodeSwapping, CantSwap: SwappingProcedure;
  SwappingProcedure: TYPE = PROCEDURE RETURNS [BOOLEAN];

  SwapStrategy: TYPE = RECORD [
    link: POINTER TO SwapStrategy,
    proc: SwappingProcedure];

  AddSwapStrategy: PROCEDURE [strategy:POINTER TO SwapStrategy];
  RemoveSwapStrategy: PROCEDURE [strategy:POINTER TO SwapStrategy];
```

```
  -- Files:

  FileObject: TYPE = File Object;
  FileHandle: TYPE = POINTER TO FileObject;

  AccessOptions: TYPE = [0..7];
    Read: AccessOptions = 1;
    Write: AccessOptions = 2;
    Append: AccessOptions = 4;
  -- Delete: AccessOptions = 8;

  VersionOptions: TYPE = [0..3];
    NewFileOnly: VersionOptions = 1;
    OldFileOnly: VersionOptions = 2;

  DefaultAccess: AccessOptions = 0;
  DefaultVersion: VersionOptions = 0;

  FileNameError: SIGNAL [name:STRING];
  InvalidFP: SIGNAL [fp:POINTER TO FP];
  FileError, FileAccessError: SIGNAL [file:FileHandle];

  NewFile: PROCEDURE [name:STRING, access:AccessOptions, version:VersionOptions] RETURNS [FileHandle];
  InsertFile: PROCEDURE [fp:POINTER TO FP, access:AccessOptions] RETURNS [FileHandle];
  OpenFile, CloseFile: PROCEDURE [file:FileHandle];
  LockFile, UnlockFile: PROCEDURE [file:FileHandle];
  ReleaseFile, DestroyFile: PROCEDURE [file:FileHandle];

  GetFileAccess: PROCEDURE [file:FileHandle] RETURNS [access:AccessOptions];
  SetFileAccess: PROCEDURE [file:FileHandle, access:AccessOptions];

  GetEndOfFile: PROCEDURE [file:FileHandle] RETURNS [page:PageNumber, byte:CARDINAL];
  SetEndOfFile: PROCEDURE [file:FileHandle, page:PageNumber, byte:CARDINAL];
  UpdateFileLength: PROCEDURE [file: FileHandle, fa: POINTER TO FA];

  JumpToPage: PROCEDURE [
    cfa:POINTER TO CFA, page:PageNumber, buf:POINTER]
    RETURNS [prev,next:vDA];

  GetFileFP: PROCEDURE [file:FileHandle, fp:POINTER TO FP];
  FindFile: PROCEDURE [fp:POINTER TO FP] RETURNS [FileHandle];

  EnumerateFiles: PROCEDURE [
    proc: PROCEDURE [FileHandle] RETURNS [BOOLEAN]]
    RETURNS [file:FileHandle];

  END.
```