```
-- MiniNub.Mesa
-- Edited by Sandman on May 19, 1978  4:24 PM

DIRECTORY
  AltoFileDefs: FROM "altofiledefs" USING [CFA, eofDA, FA],
  ControlDefs: FROM "controldefs" USING [
    GetReturnLink, GlobalFrameHandle, NullGlobalFrame, StateVector],
  CoreSwapDefs: FROM "coreswapdefs" USING [CAbort, CoreSwap],
  ImageDefs: FROM "imagedefs" USING [AddFileRequest, FileRequest],
  IODefs: FROM "iodefs" USING [CR, SP],
  LoaderDefs: FROM "loaderdefs" USING [Load, Loader, New, VersionMismatch],
  MiscDefs: FROM "miscdefs",
  ProcessDefs: FROM "processdefs" USING [Aborted],
  SegmentDefs: FROM "segmentdefs" USING [
    FileSegmentHandle, LockFile, Read, ReleaseFile, UnlockFile],
  StreamDefs: FROM "streamdefs" USING [
    CreateByteStream, GetFA, JumpToFA, Read, StreamHandle],
  StringDefs: FROM "stringdefs" USING [
    AppendChar, AppendString, EquivalentString];

MiniNub: PROGRAM
  IMPORTS CoreSwapDefs, LoaderDefs, ProcessDefs, SegmentDefs, StreamDefs, StringDefs
  EXPORTS MiscDefs =

BEGIN

CallDebugger: PROCEDURE =
  BEGIN -- user's entry point to debugger
  state: ControlDefs.StateVector;
  state.stkptr ← 0;
  state.dest ← ControlDefs.GetReturnLink[];
  CoreSwapDefs.CoreSwap[explicitcall, @state];
  RETURN
  END;

BadFile: SIGNAL [name: STRING] = CODE;

LoadNew: PROCEDURE [name: STRING, framelinks: BOOLEAN] RETURNS [PROGRAM] =
  BEGIN
  g: ControlDefs.GlobalFrameHandle;
  bcd: SegmentDefs.FileSegmentHandle;
  bcd ← LoaderDefs.Load[name
    ! BadFile, UNWIND => NULL; ANY => ERROR BadFile[name]];
  g ← LoaderDefs.New[bcd, framelinks, FALSE
    ! BadFile, LoaderDefs.VersionMismatch, UNWIND => NULL;
      ANY => ERROR BadFile[name]];
  RETURN[LOOPHOLE[g]]
  END;

comcmRequest: short ImageDefs.FileRequest ← [
  body: short[fill:, name: "Com.Cm."],
  file: NIL,
  access: SegmentDefs.Read,
  link: ];

Done: SIGNAL = CODE;

debug, start, command, framelinks: BOOLEAN;

ProcessSwitches: PROCEDURE [s: STRING] =
  BEGIN
  i: CARDINAL;
  inverse: {this, next, no} ← no;
  FOR i IN [0..s.length) DO
    SELECT s[i] FROM
      'd, 'D => debug ← inverse # this;
      's, 'S => start ← inverse # this;
      'c, 'C => command ← inverse # this;
      'l, 'L => framelinks ← ~(inverse # this);
      '- => inverse ← next;
      ENDCASE;
    inverse ← IF inverse = next THEN this ELSE no;
    ENDLOOP;
  END;

InitSwitches: PROCEDURE =
```

```
  BEGIN command ← debug ← FALSE; framelinks ← start ← TRUE; END;

GetToken: PROCEDURE [com: StreamDefs.StreamHandle, token, ext, switches: STRING] =
  BEGIN
  s: STRING;
  c: CHARACTER;
  token.length ← ext.length ← switches.length ← 0;
  s ← token;
  WHILE ~com.endof[com] DO
    SELECT (c ← com.get[com]) FROM
      IODefs.SP, IODefs.CR =>
        IF token.length # 0 OR ext.length # 0 OR switches.length # 0 THEN RETURN;
      '. => s ← ext;
      '/ => s ← switches;
      ENDCASE => StringDefs.AppendChar[s, c];
    ENDLOOP;
  RETURN
  END;

cfa: AltoFileDefs.CFA;

CommandLineCFA: PUBLIC PROCEDURE RETURNS [POINTER TO AltoFileDefs.CFA] =
  BEGIN
  RETURN[@cfa]
  END;

LoadSystems: PROCEDURE =
  BEGIN
  user: PROGRAM;
  p: PROCESS;
  IF comcmRequest.file = NIL THEN
    BEGIN cfa.fp ← [[1,0,1,17777B,177777B],AltoFileDefs.eofDA]; RETURN END;
  cfa.fp ← comcmRequest.file.fp;
  SegmentDefs.LockFile[comcmRequest.file];
  InitSwitches[];
  SkipImage[@cfa.fa];
  IF debug THEN CallDebugger[];
  DO
    InitSwitches[];
    user ← LoadUser[@cfa.fa ! Done => EXIT];
    IF debug THEN CallDebugger[];
    IF start AND user # LOOPHOLE[ControlDefs.NullGlobalFrame] THEN
      BEGIN
      p ← FORK StartModule[user];
      JOIN p;
      END;
    ENDLOOP;
  SegmentDefs.UnlockFile[comcmRequest.file];
  SegmentDefs.ReleaseFile[comcmRequest.file];
  END;

StartModule: PROCEDURE [f: PROGRAM] =
  BEGIN
  START f [! ProcessDefs.Aborted, CoreSwapDefs.CAbort => CONTINUE];
  RETURN
  END;

LoadUser: PROCEDURE [fa: POINTER TO AltoFileDefs.FA] RETURNS [user: PROGRAM] =
  BEGIN OPEN IODefs, StreamDefs;
  com: StreamHandle;
  name: STRING ← [40];
  ext: STRING ← [10];
  switches: STRING ← [5];
  com ← CreateByteStream[comcmRequest.file, Read];
  BEGIN
    StreamDefs.JumpToFA[com, fa ! ANY => GO TO finished];
    GetToken[com, name, ext, switches];
    IF name.length = 0 THEN GO TO finished;
    IF ext.length = 0 THEN ext ← "bcd";
    StringDefs.AppendChar[name, '.];
    StringDefs.AppendString[name, ext];
    StreamDefs.GetFA[com, fa];
    com.destroy[com];
    ProcessSwitches[switches];
    IF command THEN
      BEGIN
```

```
      ProcessSwitches[name];
      user ← LOOPHOLE[ControlDefs.NullGlobalFrame];
      END
    ELSE user ← LoadNew[name, framelinks];
  EXITS
    finished => BEGIN com.destroy[com]; SIGNAL Done; END;
  END;
  END;

SkipImage: PROCEDURE [fa: POINTER TO AltoFileDefs.FA] =
  BEGIN OPEN IODefs, StreamDefs;
  com: StreamHandle;
  name: STRING ← [40];
  ext: STRING ← [10];
  switches: STRING ← [5];
  com ← CreateByteStream[comcmRequest.file, Read];
  StreamDefs.GetFA[com, fa];
  GetToken[com, name, ext, switches];
  IF StringDefs.EquivalentString[ext, "image"] THEN StreamDefs.GetFA[com, fa];
  com.destroy[com];
  ProcessSwitches[switches];
  END;

-- Main body

ImageDefs.AddFileRequest[@comcmRequest];

STOP;

START LoaderDefs.Loader; -- make Loader fill in SD
LoadSystems[! CoreSwapDefs.CAbort => CONTINUE];

END...
```