```
-- NubControl.Mesa
-- Edited by Johnsson on July 26, 1978  8:29 AM

DIRECTORY
  AltoFileDefs: FROM "altofiledefs" USING [CFA, eofDA, FA],
  ControlDefs: FROM "controldefs" USING [
    GFT, GlobalFrameHandle, NullGlobalFrame],
  CoreSwapDefs: FROM "coreswapdefs" USING [CAbort, CantSwap],
  DisplayDefs: FROM "displaydefs" USING [DisplayControl],
  ImageDefs: FROM "imagedefs" USING [
    AddFileRequest, FileRequest, ImageVersion, MakeImage, StopMesa],
  IODefs: FROM "iodefs" USING [
    CR, DEL, LineOverflow, NewLine, NUL, ReadChar, ReadID, ReadNumber,
    Rubout, SP, WriteChar, WriteOctal, WriteString],
  LoaderDefs: FROM "loaderdefs" USING [Load, Loader, New, VersionMismatch],
  LoaderUtilityDefs: FROM "loaderutilitydefs" USING [FileNotFound],
  MiscDefs: FROM "miscdefs" USING [CallDebugger],
  ProcessDefs: FROM "processdefs" USING [Aborted],
  SegmentDefs: FROM "segmentdefs" USING [
    FileSegmentHandle, LockFile, Read, ReleaseFile, UnlockFile],
  StreamDefs: FROM "streamdefs" USING [
    CreateByteStream, GetFA, JumpToFA, KeyStreams, Read, StreamHandle],
  StringDefs: FROM "stringdefs" USING [
    AppendChar, AppendString, EquivalentString, InvalidNumber,
    StringBoundsFault],
  TimeDefs: FROM "timedefs" USING [AppendDayTime, DefaultTime, UnpackDT];

NubControl: PROGRAM
  IMPORTS CoreSwapDefs, DisplayDefs, ImageDefs, IODefs, LoaderDefs,
    ProcessDefs, SegmentDefs, StreamDefs, StringDefs, TimeDefs,
    LoaderUtilityDefs
  EXPORTS MiscDefs =

BEGIN

-- System Signals are converted to these to prevent NubCommand
-- from catching user generated signals
Delete: SIGNAL = CODE;          -- nee Rubout
StringTooLong: ERROR = CODE;    -- nee LineOverflow
BadFile: ERROR [badname: STRING] = CODE;        -- nee FileNameError
BadVersion: SIGNAL [badname: STRING] = CODE;    -- nee VersionMismatch
BadNumber: ERROR = CODE;                -- nee InvalidNumber

GlobalFrameHandle: TYPE = ControlDefs.GlobalFrameHandle;
NullGlobalFrame: GlobalFrameHandle = ControlDefs.NullGlobalFrame;

useCommandLine, skipImage: BOOLEAN ← TRUE;
idstring: STRING ← [40];
defaultframe: GlobalFrameHandle ← NullGlobalFrame;

DoCommand: PROCEDURE =
  BEGIN OPEN ControlDefs;
  p: PROCESS;
  f: GlobalFrameHandle;
  IF skipImage THEN SkipImage[];
  f ← IF useCommandLine THEN LoadSystem[] ELSE Command[];
  IF f # NullGlobalFrame THEN
    BEGIN
    p ← FORK StartModule[LOOPHOLE[f]];
    JOIN p;
    END;
  RETURN
  END;

StartModule: PROCEDURE [f: PROGRAM] =
  BEGIN
  BEGIN
  ENABLE ProcessDefs.Aborted, CoreSwapDefs.CAbort => CONTINUE;
  IF ~LOOPHOLE[f, GlobalFrameHandle].started THEN START f ELSE RESTART f;
  END;
  RETURN
  END;

Command: PROCEDURE RETURNS [GlobalFrameHandle] =
  BEGIN OPEN IODefs;
  nCommand: STRING = "New"L;
```

```
  sCommand: STRING = "Start"L;
  mCommand: STRING = "MakeImage"L;
  dCommand: STRING = "Debug"L;
  qCommand: STRING = "Quit"L;
  c: CHARACTER;
  f: GlobalFrameHandle;
  DO
  WriteEOL[]; WriteChar['>];
    SELECT c←ReadChar[] FROM
      'N,'n =>
        BEGIN
        WriteString[nCommand];
        LoadModule[];
        END;
      'S,'s =>
        BEGIN
        WriteString[sCommand];
        f ← getgframe[];
        WriteEOL[];
        RETURN[f];
        END;
      'D,'d =>
        BEGIN
        WriteString[dCommand];
        confirm[];
        MiscDefs.CallDebugger[NIL];
        END;
      'Q,'q =>
        BEGIN
        WriteString[qCommand];
        confirm[];
        ImageDefs.StopMesa[];
        END;
      'M,'m =>
        BEGIN
        WriteString[mCommand];
        ImageDefs.MakeImage[getfilename[".image"L]];
        END;
      CR,SP => NULL;
      ENDCASE =>
        BEGIN
        WriteChar[c];
        WriteString[" Commands are:"L];
        WriteChar[' ]; WriteString[nCommand];
        WriteChar[' ]; WriteString[sCommand];
        WriteChar[' ]; WriteString[mCommand];
        WriteChar[' ]; WriteString[dCommand];
        WriteChar[' ]; WriteString[qCommand];
        END;
    ENDLOOP;
  RETURN[NullGlobalFrame]
  END;

LoadModule: PROCEDURE =
  BEGIN
  name: STRING ← [40];
  ext: STRING ← [10];
  switches: STRING ← [10];
  i: CARDINAL ← 0;
  g: GlobalFrameHandle;
  fn: STRING = " Filename: "L;
  get: PROCEDURE RETURNS [c: CHARACTER] =
    BEGIN
    IF i = idstring.length THEN RETURN[IODefs.NUL];
    c ← idstring[i];
    i ← i + 1;
    RETURN[c]
    END;
  IODefs.WriteString[fn];
  IODefs.ReadID[idstring ! IODefs.Rubout => ERROR Delete;
    IODefs.LineOverflow => ERROR StringTooLong];
  GetToken[get, name, ext, switches];
  IF ext.length = 0 THEN ext ← "bcd";
  StringDefs.AppendChar[name, '.];
  StringDefs.AppendString[name, ext];
  InitSwitches[];
```

```
  ProcessSwitches[switches];
  g ← LoadNew[name, framelinks];
  IF g # NullGlobalFrame THEN defaultframe ← g;
  RETURN
  END;

LoadNew: PROCEDURE [name: STRING, framelinks: BOOLEAN]
  RETURNS [g: GlobalFrameHandle] =
  BEGIN OPEN LoaderDefs;
  s: STRING = " -- "L;
  bcd: SegmentDefs.FileSegmentHandle;
  bcd ← LoaderDefs.Load[name
    ! BadFile, UNWIND => NULL; ANY => ERROR BadFile[name]];
  g ← LoaderDefs.New[bcd, framelinks, FALSE
    ! BadFile, BadVersion, UNWIND => NULL;
      LoaderDefs.VersionMismatch =>
        BEGIN SIGNAL BadVersion[name]; RESUME END;
      LoaderUtilityDefs.FileNotFound => ERROR BadFile[name];
      ANY => ERROR BadFile[name]];
  IODefs.WriteString[s];
  IODefs.WriteOctal[g];
  WriteEOL[];
  RETURN
  END;

getgframe: PROCEDURE RETURNS [f: GlobalFrameHandle] =
  BEGIN OPEN ControlDefs, IODefs;
  gf: STRING = " Global frame: "L;
  ngf: STRING = " not a global frame!"L;
  WriteString[gf];
  f ← ReadNumber[defaultframe,8
    ! LineOverflow => ERROR StringTooLong;
      Rubout => ERROR Delete;
      StringDefs.InvalidNumber => ERROR BadNumber];
  IF GFT[f.gfi].frame # f THEN
    BEGIN WriteString[ngf]; SIGNAL CoreSwapDefs.CAbort END;
  defaultframe ← f;
  RETURN
  END;


getfilename: PROCEDURE [defaultextension: STRING] RETURNS [STRING] =
  BEGIN OPEN IODefs;
  fn: STRING = " Filename: "L;
  WriteString[fn];
  ReadID[idstring ! Rubout => ERROR Delete;
    LineOverflow => ERROR StringTooLong];
  defaultext[idstring, defaultextension];
  RETURN[idstring]
  END;

defaultext: PROCEDURE [idstring: STRING, ext: STRING] =
  BEGIN
  i: CARDINAL;
  FOR i IN [0..idstring.length) DO
    IF idstring[i] = '. THEN EXIT;
    REPEAT FINISHED =>
      FOR i IN [0..ext.length) DO
        StringDefs.AppendChar[idstring, ext[i]
          ! StringDefs.StringBoundsFault => ERROR StringTooLong];
        ENDLOOP;
    ENDLOOP;
  END;

confirm: PROCEDURE =
  BEGIN OPEN IODefs;
  c: STRING = " [Confirm] "L;
  WriteString[c];
  DO
    SELECT ReadChar[] FROM
      CR => EXIT;
      DEL => ERROR Delete;
      ENDCASE => WriteChar['?];
    ENDLOOP;
  WriteChar[CR];
  RETURN
```

```
  END;


WriteEOL: PROCEDURE =
  BEGIN OPEN IODefs;
  IF ~NewLine[] THEN WriteChar[CR];
  RETURN
  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: BOOLEAN ← FALSE;
  FOR i IN [0..s.length) DO
    SELECT s[i] FROM
      'c, 'C => BEGIN inverse ← FALSE; command ← TRUE END;
      'd, 'D => BEGIN inverse ← FALSE; debug ← TRUE END;
      's, 'S => IF inverse THEN inverse ← start ← FALSE ELSE start ← TRUE;
      'l, 'L => BEGIN inverse ← FALSE; framelinks ← FALSE END;
      '- => inverse ← TRUE;
      ENDCASE => inverse ← FALSE;
    ENDLOOP;
  END;

InitSwitches: PROCEDURE =
  BEGIN command ← debug ← FALSE; framelinks ← start ← TRUE; END;

GetToken: PROCEDURE [
  get: PROCEDURE RETURNS [CHARACTER], token, ext, switches: STRING] =
  BEGIN OPEN IODefs;
  s: STRING;
  c: CHARACTER;
  token.length ← ext.length ← switches.length ← 0;
  s ← token;
  WHILE (c ← get[]) # NUL DO
    SELECT c FROM
      SP, CR =>
        IF token.length # 0 OR ext.length # 0 OR switches.length # 0 THEN RETURN;
      '. => s ← ext;
      '/ => s ← switches;
      ENDCASE => StringDefs.AppendChar[s, c
        ! StringDefs.StringBoundsFault => ERROR StringTooLong];
    ENDLOOP;
  RETURN
  END;

cfa: AltoFileDefs.CFA;

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

CleanUpCommandLine: PROCEDURE =
  BEGIN
  SegmentDefs.UnlockFile[comcmRequest.file];
  SegmentDefs.ReleaseFile[comcmRequest.file];
  comcmRequest.file ← NIL;
  useCommandLine ← FALSE;
  RETURN
  END;

LoadSystem: PROCEDURE RETURNS [user: GlobalFrameHandle] =
  BEGIN
  BEGIN
    ENABLE UNWIND => CleanUpCommandLine[];
```

```
    InitSwitches[];
    user ← LoadUser[@cfa.fa ! Done => GOTO done];
    IF user # NullGlobalFrame THEN defaultframe ← user;
    IF debug THEN MiscDefs.CallDebugger[NIL];
    IF ~start THEN user ← NullGlobalFrame;
  EXITS
    done =>
      BEGIN user ← NullGlobalFrame; CleanUpCommandLine[] END;
  END;
  RETURN
  END;

LoadUser: PROCEDURE [fa: POINTER TO AltoFileDefs.FA]
  RETURNS [user: GlobalFrameHandle] =
  BEGIN OPEN IODefs, StreamDefs;
  com: StreamHandle;
  name: STRING ← [40];
  ext: STRING ← [10];
  switches: STRING ← [10];
  get: PROCEDURE RETURNS [c: CHARACTER] =
    BEGIN
    IF com.endof[com] THEN RETURN[NUL];
    RETURN[com.get[com]];
    END;
  com ← CreateByteStream[comcmRequest.file, Read];
  BEGIN
    StreamDefs.JumpToFA[com, fa ! ANY => GO TO finished];
    GetToken[get, name, ext, switches];
    IF name.length = 0 THEN GO TO finished;
    IF ext.length = 0 THEN ext ← "bcd"L;
    WriteEOL[]; WriteChar['>];
    WriteString[name];
    StringDefs.AppendChar[name, '.];
    StringDefs.AppendString[name, ext];
    StreamDefs.GetFA[com, fa];
    com.destroy[com];
    ProcessSwitches[switches];
    IF command THEN
      BEGIN
      ProcessSwitches[name];
      user ← NullGlobalFrame;
      END
    ELSE user ← LoadNew[name, framelinks];
  EXITS
    finished => BEGIN com.destroy[com]; SIGNAL Done; END;
  END;
  RETURN
  END;

SkipImage: PROCEDURE =
  BEGIN OPEN IODefs, StreamDefs;
  com: StreamHandle;
  name: STRING ← [40];
  ext: STRING ← [10];
  switches: STRING ← [10];
  get: PROCEDURE RETURNS [c: CHARACTER] =
    BEGIN
    IF com.endof[com] THEN RETURN[NUL];
    RETURN[com.get[com]];
    END;
  skipImage ← FALSE;
  IF comcmRequest.file = NIL THEN
    BEGIN
    cfa.fp ← [[1,0,1,17777B,177777B],AltoFileDefs.eofDA];
    useCommandLine ← FALSE;
    RETURN
    END;
  cfa.fp ← comcmRequest.file.fp;
  SegmentDefs.LockFile[comcmRequest.file];
  InitSwitches[];
  com ← CreateByteStream[comcmRequest.file, Read];
  StreamDefs.GetFA[com, @cfa.fa];
  GetToken[get, name, ext, switches];
  IF StringDefs.EquivalentString[ext, "image"L] THEN
    StreamDefs.GetFA[com, @cfa.fa];
  com.destroy[com];
```

```
  ProcessSwitches[switches];
  IF debug THEN MiscDefs.CallDebugger[NIL];
  END;

WriteHerald: PROCEDURE =
  BEGIN OPEN TimeDefs;
  h: STRING = "Alto/Mesa 4.1 of "L;
  time: STRING ← [18];
  AppendDayTime[time, UnpackDT[ImageDefs.ImageVersion[].time]];
  time.length ← time.length - 3;
  IODefs.WriteString[h]; IODefs.WriteString[time];
  time.length ← 0;
  AppendDayTime[time, UnpackDT[DefaultTime]];
  time.length ← time.length - 3;
  WriteEOL[]; IODefs.WriteString[time];
  END;

ErrorName: TYPE = {XXX, file, number, toolong, nodebug, aborted, diffver};

WriteError: PROCEDURE [error: ErrorName] =
  BEGIN
  IODefs.WriteString[SELECT error FROM
    file => "!File: "L,
    number => "!Number"L,
    toolong => "!String too long"L,
    nodebug =>
      "External Debugger not installed, type DEL to abort "L,
    aborted => "...Aborted..."L,
    diffver => " referenced in different versions"L,
    ENDCASE => " XXX"L]
  END;


-- Main body

START StreamDefs.KeyStreams; -- Start keyboard process

ImageDefs.AddFileRequest[@comcmRequest];
START DisplayDefs.DisplayControl;

STOP;

RESTART DisplayDefs.DisplayControl;
START LoaderDefs.Loader;

WriteHerald[];

DO OPEN IODefs;  -- forever
  DoCommand[
    ! Delete => BEGIN WriteError[XXX]; CONTINUE END;
      CoreSwapDefs.CAbort => CONTINUE;
      BadFile --[badname: STRING]-- =>
        BEGIN
        WriteEOL[];
        WriteError[file];
        WriteString[badname];
        CONTINUE
        END;
      BadNumber =>
        BEGIN WriteEOL[]; WriteError[number]; CONTINUE END;
      StringTooLong =>
        BEGIN WriteEOL[]; WriteError[toolong]; CONTINUE END;
      CoreSwapDefs.CantSwap =>
        BEGIN
        WriteEOL[];
        WriteError[nodebug];
        UNTIL ReadChar[]=DEL DO WriteChar['?] ENDLOOP;
        WriteError[aborted];
        CONTINUE
        END;
      BadVersion --[badname: STRING]-- =>
        BEGIN
        WriteEOL[];
        WriteError[file];
        WriteString[badname];
        WriteError[diffver];
```

```
        RESUME
        END;
      UNWIND =>CONTINUE
    ];
  ENDLOOP;


END..
```