```
-- File Process.Mesa
-- Last edited by Sandman on August 23, 1977  9:49 PM

DIRECTORY
  FrameDefs: FROM "framedefs",
  InlineDefs: FROM "inlinedefs",
  SystemDefs: FROM "systemdefs",
  ControlDefs: FROM "controldefs",
  ProcessDefs: FROM "processdefs";

DEFINITIONS FROM
  InlineDefs, ControlDefs, ProcessDefs;

Process: PROGRAM IMPORTS FrameDefs, SystemDefs EXPORTS ProcessDefs SHARES ProcessDefs =
BEGIN

PVector: POINTER TO ProcessVector ← PV;

PriorityNotAvailable: PUBLIC ERROR = CODE;
InvalidPriority: PUBLIC ERROR = CODE;
InvalidProcess: PUBLIC ERROR = CODE;
ProcessNotScheduled: PUBLIC ERROR = CODE;

ValidatePriority: PROCEDURE[p: ProcessPriority] =
  BEGIN
    IF p = Unscheduled OR
      p IN [HighestProcessPriority..LowestProcessPriority] THEN RETURN;
    ERROR InvalidPriority;
  END;


ValidateProcess: PROCEDURE[p:ProcessHandle] =
  BEGIN
    ValidatePriority[p.pn ! InvalidPriority => ERROR InvalidProcess ];
    IF p.pn = Unscheduled THEN RETURN;
    IF PVector[p.pn] # p THEN ERROR InvalidProcess;
  END;


CreateProcessFromFrame: PUBLIC PROCEDURE [frame: FrameHandle, priority: ProcessPriority]
  RETURNS [p: ProcessHandle] =
  BEGIN OPEN p.state;
    ValidatePriority[priority];
    IF PVector[priority] # ProcessNIL THEN ERROR PriorityNotAvailable;

    p←SystemDefs.AllocateHeapNode[SIZE[ProcessObject]];
    instbyte←0;
    IF FrameDefs.Class[frame] = global THEN
      BEGIN
      stkptr ← 1;
      stk[0] ← 0;
      END
    ELSE stkptr ← 0;
    X ← frame;
    Y ← 0;
    p.pn ← priority;
    IF priority # Unscheduled THEN PVector[priority] ← p;
  END;


CreateProcessFromProcedure: PUBLIC PROCEDURE [proc: PROCEDURE, priority: ProcessPriority]
  RETURNS [p: ProcessHandle] =
  BEGIN OPEN p.state;
    ValidatePriority[priority];
    IF PVector[priority] # ProcessNIL THEN ERROR PriorityNotAvailable;

    p←SystemDefs.AllocateHeapNode[SIZE[ProcessObject]];
    instbyte←0;
    stkptr←0;
    X←proc;
    Y←0;
    p.pn←priority;
    IF priority # Unscheduled THEN PVector[priority]←p;
  END;

DestroyProcess: PUBLIC PROCEDURE [p: ProcessHandle] =
  BEGIN
    ValidateProcess[p];
    SetProcessPriority[p, Unscheduled];
    SystemDefs.FreeHeapNode[p];
  END;


MakeProcessReady: PUBLIC PROCEDURE [p: ProcessHandle] =
  BEGIN
    ValidateProcess[p];
    IF p.pn = Unscheduled THEN ERROR ProcessNotScheduled;
    DisableInterrupts[];
    RP↑ ← BITOR[RP↑,BITSHIFT[1,p.pn]];
    WakeupsWaiting↑ ← BITOR[WakeupsWaiting↑,BITSHIFT[1,p.pn]];
    EnableInterrupts[];
    RETURN
  END;


ActivateProcess: PUBLIC PROCEDURE [p: ProcessHandle] =
  BEGIN
    ValidateProcess[p];
    IF p.pn = Unscheduled THEN ERROR ProcessNotScheduled;
    DisableInterrupts[];
    AP↑ ← BITOR[AP↑,BITSHIFT[1,p.pn]];
    ActiveWord↑ ← BITOR[ActiveWord↑,BITSHIFT[1,p.pn]];
    EnableInterrupts[];
    RETURN
  END;


DeActivateProcess: PUBLIC PROCEDURE [p: ProcessHandle] =
  BEGIN
    ValidateProcess[p];
    IF p.pn = Unscheduled THEN ERROR ProcessNotScheduled;
    DisableInterrupts[];
    AP↑ ← BITAND[AP↑,BITNOT[BITSHIFT[1,p.pn]]];
    ActiveWord↑ ← BITAND[ActiveWord↑,BITNOT[BITSHIFT[1,p.pn]]];
    EnableInterrupts[];
    RETURN
  END;


EnumerateProcess: PUBLIC PROCEDURE [proc: PROCEDURE[ProcessHandle] RETURNS [BOOLEAN]]
    RETURNS [p: ProcessHandle] =
  BEGIN
    i: ProcessPriority;
    FOR i IN [HighestProcessPriority..LowestProcessPriority] DO
        IF (p←PVector[i]) # ProcessNIL AND p # NIL THEN
            IF proc[p] THEN RETURN;
        ENDLOOP;
    RETURN [ProcessNIL]
  END;


SetProcessPriority: PUBLIC PROCEDURE [p: ProcessHandle, priority: ProcessPriority] =
  BEGIN
    m: WORD;
    ValidateProcess[p];
    IF PVector[priority] # ProcessNIL THEN ERROR PriorityNotAvailable;
    IF p.pn # Unscheduled THEN
        BEGIN
        m ← BITNOT[BITSHIFT[1,p.pn]];
        DisableInterrupts[];
        RP↑ ← BITAND[RP↑,m];
        WakeupsWaiting↑ ← BITAND[WakeupsWaiting↑,m];
        AP↑ ← BITAND[AP↑,m];
        ActiveWord↑ ← BITAND[ActiveWord↑,m];
        PVector[p.pn] ← ProcessNIL;
        EnableInterrupts[];
        END;
    p.pn←priority;
    IF priority # Unscheduled THEN PVector[priority] ← p;
  END;

GetProcessPriority: PUBLIC PROCEDURE [p: ProcessHandle] RETURNS [ProcessPriority] =
  BEGIN
    ValidateProcess[p];
    RETURN [p.pn]
  END;

GetCurrentProcess: PUBLIC PROCEDURE RETURNS [ProcessHandle] =
  BEGIN
    RETURN [PVector[CPN↑]]
  END;

GetCurrentPriority: PUBLIC PROCEDURE RETURNS [ProcessPriority] =
  BEGIN
    RETURN [CPN↑]
  END;


END...
```