```
-- DUserProc.Mesa
-- Edited by:
--           Sandman, April 15, 1978  3:35 PM
--           Jerry Morrison, March 29, 1978  2:57 PM
--           Barbara, May 12, 1978  4:21 PM
--           Johnsson, August 30, 1978  10:51 AM
--    Checks on ControlDELtyped before calling a userproc
--    Allows the user to type any unique partial userproc name

DIRECTORY
  DebugUsefulDefs: FROM "debugusefuldefs",
  DebugUtilityDefs: FROM "debugutilitydefs",
  IODefs: FROM "iodefs" USING [CR, ReadEditedString, Rubout, WriteChar, WriteLine, WriteString],
  StreamDefs: FROM "streamdefs" USING [ControlDELtyped],
  StringDefs: FROM "stringdefs" USING [AppendString, EquivalentString, EquivalentSubStrings, SubString,
** SubStringDescriptor],
  SystemDefs: FROM "systemdefs" USING [AllocateHeapNode, AllocateHeapString];

DUserProc: PROGRAM
  IMPORTS IODefs, StreamDefs, StringDefs, SystemDefs
  EXPORTS DebugUsefulDefs, DebugUtilityDefs =

BEGIN

AmbiguousTag: PUBLIC SIGNAL [oldTag: STRING] = CODE;

ProcHandle: TYPE = POINTER TO ProcItem;
ProcItem: TYPE = RECORD [
  link: ProcHandle,
  item: STRING,
  proc: PROCEDURE];

head: ProcHandle ← NIL;

AddCommand: PUBLIC PROCEDURE [tag: STRING, proc: PROCEDURE] =
  BEGIN
  p, new: ProcHandle;
  IF (new ← Unique[tag]) # NIL THEN
    BEGIN
    SIGNAL AmbiguousTag[tag];
    new.proc ← proc;
    RETURN
    END;
  new ← SystemDefs.AllocateHeapNode[SIZE[ProcItem]];
  new.item ← SystemDefs.AllocateHeapString[tag.length];
  new.item.length ← 0;
  StringDefs.AppendString[to: new.item, from: tag];
  new.proc ← proc;
  new.link ← NIL;
  IF head = NIL THEN head ← new
  ELSE
    FOR p ← head, p.link DO
      IF p.link = NIL THEN BEGIN p.link ← new; EXIT END;
      ENDLOOP;
  RETURN
  END;  -- Of AddCommand

NoUserProcsLoaded: PUBLIC SIGNAL = CODE;

UserProc: PUBLIC PROCEDURE =
  BEGIN OPEN IODefs;
  i: ProcHandle;
  s: STRING ← [40];
  prompt: STRING = "Proc: "L;

  eol: PROCEDURE [c: CHARACTER] RETURNS [BOOLEAN] =
    BEGIN
    IF s.length > 0 THEN RETURN[c = CR];
    IF c = '? THEN BEGIN WriteChar[c]; RETURN[TRUE] END;
    RETURN [c = CR]
    END;

  IF head = NIL THEN BEGIN SIGNAL NoUserProcsLoaded; RETURN END;
  IF head.link = NIL THEN
    BEGIN
    WriteString[prompt];
    IF StreamDefs.ControlDELtyped[] THEN
      BEGIN WriteLine[" ... aborted"L]; RETURN END;
    WriteLine[head.item];
    head.proc[];
    RETURN;
    END;
  DO
    WriteString[prompt];
    [] ← ReadEditedString[s, eol, TRUE ! Rubout => GOTO del];
    IF StreamDefs.ControlDELtyped[] THEN GOTO aborted;
    WriteChar[CR];
    IF (i ← Match[s]) # NIL THEN BEGIN i.proc[]; RETURN; END;
    FOR i ← head, i.link UNTIL i = NIL DO
      IF StreamDefs.ControlDELtyped[] THEN GOTO aborted;
      WriteLine[i.item];
      ENDLOOP;
    REPEAT
      del => WriteLine[" XXX"L];
      aborted => WriteLine[" ... aborted"L];
    ENDLOOP;
  END;  -- Of CallUserProc

--Unique scans the list of installed userprocs for an exact match with s
Unique: PROCEDURE [s: STRING] RETURNS [pr: ProcHandle] =
  BEGIN
  FOR pr ← head, pr.link UNTIL pr = NIL DO
    IF StringDefs.EquivalentString[pr.item,s] THEN RETURN
    ENDLOOP;
  RETURN[NIL]
  END;  -- Of Unique

-- Match scans the list of userprocs for any unique name specification
Match: PROCEDURE [name: STRING] RETURNS [pr0: ProcHandle] =
  BEGIN
  pr1: ProcHandle;
  IF (pr0 ← Unique[name]) # NIL THEN RETURN; -- first check for exact match
  -- that failed; so scan for a unique partial specification
  FOR pr0 ← head, pr0.link UNTIL pr0 = NIL DO
    IF MatchFirstPart[name, pr0.item] THEN
      -- the specified name matches a Userproc name; make sure it's unique
      FOR pr1 ← pr0.link, pr1.link UNTIL pr1 = NIL DO
        IF MatchFirstPart[name, pr1.item] THEN RETURN [NIL] -- not unique
        REPEAT
          FINISHED => RETURN;  -- name is a unique partial spec
        ENDLOOP;
    ENDLOOP;
  RETURN [NIL]  -- didn't find name on either scan
  END;  -- of Match

-- MatchFirstPart returns TRUE iff short matches the first part of long
MatchFirstPart: PROCEDURE [short, long: STRING] RETURNS [BOOLEAN] =
  BEGIN OPEN StringDefs;
  shortSSD: SubStringDescriptor ← [short, 0, short.length];
  longSSD: SubStringDescriptor ← [long, 0, short.length];
  shortSS: SubString ← @shortSSD;
  longSS: SubString ← @longSSD;
  RETURN[
    long.length > short.length AND EquivalentSubStrings[shortSS, longSS]];
  END;  -- of MatchFirstPart


END..
```