DIRECTORY
IODefs: FROM "iodefs",
AltoDefs: FROM "altodefs",
SegmentDefs: FROM "segmentdefs";

DEFINITIONS FROM AltoDefs, IODefs, SegmentDefs;

SegMap: PROGRAM IMPORTS IODefs, SegmentDefs SHARES SegmentDefs = PUBLIC BEGIN

byte: NumberFormat = NumberFormat[8,FALSE,TRUE,3];
word: NumberFormat = NumberFormat[8,FALSE,TRUE,6];

PrintDataSegment: PROCEDURE [seg:DataSegmentHandle] RETURNS [BOOLEAN] =
BEGIN OPEN seg;
WriteNumber[VMpage,byte]; WriteChar[SP];
WriteNumber[AddressFromPage[VMpage],word];
WriteString["L"];
WriteNumber[pages,byte];
WriteLine[" VM"];
WriteChar[CR];
RETURN[-More[]]
END;

PrintFileSegment: PROCEDURE[seg:FileSegmentHandle] RETURNS [BOOLEAN] =
BEGIN OPEN seg;
WriteNumber[VMpage,byte]; WriteChar[SP];
WriteNumber[AddressFromPage[VMpage],word]; WriteChar[SP];
WriteNumber[base,byte]; WriteChar[SP];
WriteString["SN"]; WriteOctal[file.fp.serial.part2];
SELECT class FROM
code => WriteString[" code"];
ENDCASE;
WITH seg SELECT FROM
remote => WriteString[" remote"]; ENDCASE;
IF read OR write THEN WriteChar['] ;
IF read THEN WriteChar['R'];
IF write THEN WriteChar['W'];
IF swappedin THEN WriteString[" in"];
IF lock > 0 THEN
BEGIN
WriteString[" lock="];
WriteOctal[lock];
END;
WriteChar[CR];
RETURN[-More[]]
END;

lc: INTEGER;
full: INTEGER = 18;

More: PROCEDURE RETURNS [BOOLEAN] =
BEGIN c: CHARACTER;
IF (lc = 0) THEN
BEGIN lc = 0;
DO -- until non-random input
SELECT (c = ReadChar[]) FROM
SP,CR,LF => EXIT;
DEL => RETURN[FALSE];
ENDCASE;
ENDLOOP;
END;
RETURN[TRUE]
END;

DO WriteChar[CR]; lc = 0;
[] + EnumerateFileSegments[PrintFileSegment];
WriteChar[CR]; lc = full;
IF More[] THEN
[] + EnumerateDataSegments[PrintDataSegment];
WriteChar[CR]; STOP;
ENDLOOP;
END.