```
-- File: Strings.Mesa  Edited by Sandman on July 22, 1977  11:02 AM

DIRECTORY
  AltoDefs: FROM "altodefs",
  InlineDefs: FROM "inlinedefs",
  StringDefs: FROM "stringdefs";

DEFINITIONS FROM AltoDefs, StringDefs;

Strings: DATA EXPORTS StringDefs SHARES StringDefs = PUBLIC
  BEGIN

  WordsForString: PROCEDURE [nchars: INTEGER] RETURNS [INTEGER] =
    BEGIN
    RETURN [StringHeaderSize + (nchars+(CharsPerWord-1))/CharsPerWord]
    END;

  StringBoundsFault: SIGNAL [s: STRING]  RETURNS [ns: STRING] = CODE;

  AppendChar: PROCEDURE [s: STRING, c: CHARACTER] =
    BEGIN
    UNTIL s.length < s.maxlength DO
      s ← SIGNAL StringBoundsFault[s];
      ENDLOOP;
    s[s.length] ← c;  s.length ← s.length+1;
    RETURN
    END;

  AppendString: PROCEDURE [to, from: STRING] =
    BEGIN
    i, j, n: CARDINAL;
    WHILE from.length + to.length > to.maxlength DO
      to ← SIGNAL StringBoundsFault[to];
      ENDLOOP;
    n ← MIN [from.length, LOOPHOLE[to.maxlength-to.length, CARDINAL]];
    i ← to.length;  j ← 0;
    WHILE j < n
      DO
      to[i] ← from[j];
      i ← i+1; j ← j+1;
      ENDLOOP;
    to.length ← i;
    RETURN
    END;

  EqualString: PROCEDURE [s1, s2: STRING] RETURNS [BOOLEAN] =
    BEGIN
    i: CARDINAL;
    IF s1.length # s2.length
      THEN RETURN [FALSE];
    FOR i IN [0..s1.length)
      DO
      IF s1[i] # s2[i]
        THEN RETURN [FALSE];
      ENDLOOP;
    RETURN [TRUE]
    END;

  EquivalentString: PROCEDURE [s1, s2: STRING] RETURNS [BOOLEAN] =
    BEGIN
    OPEN InlineDefs;
    i: CARDINAL;
    casebit: WORD = 40B;
    IF s1.length # s2.length
      THEN RETURN [FALSE];
    FOR i IN [0..s1.length)
      DO
      IF BITOR[LOOPHOLE[s1[i]], casebit] # BITOR[LOOPHOLE[s2[i]], casebit]
        THEN RETURN [FALSE];
      ENDLOOP;
    RETURN [TRUE]
    END;


  AppendSubString: PROCEDURE [to: STRING, from: SubString] =
    BEGIN
```

```
    i, j, n: CARDINAL;
    WHILE from.length + to.length > to.maxlength DO
      to ← SIGNAL StringBoundsFault[to];
      ENDLOOP;
    n ← MIN [from.length, LOOPHOLE[to.maxlength-to.length, CARDINAL]];
    i ← to.length;  j ← from.offset;
    WHILE n > 0
      DO
      to[i] ← from.base[j];
      i ← i+1; j ← j+1;  n ← n-1;
      ENDLOOP;
    to.length ← i;
    RETURN
    END;

  EqualSubStrings: PROCEDURE [s1, s2: SubString] RETURNS [BOOLEAN] =
    BEGIN
    i1, i2, n: CARDINAL;
    b1, b2: STRING;
    IF s1.length # s2.length
      THEN RETURN [FALSE];
    b1 ← s1.base;  i1 ← s1.offset;
    b2 ← s2.base;  i2 ← s2.offset;
    FOR n ← s1.length, n-1 WHILE n > 0
      DO
      IF b1[i1] # b2[i2]
        THEN RETURN [FALSE];
      i1 ← i1+1;  i2 ← i2+1;
      ENDLOOP;
    RETURN [TRUE]
    END;

  EquivalentSubStrings: PROCEDURE [s1, s2: SubString] RETURNS [BOOLEAN] =
    BEGIN
    OPEN InlineDefs;
    casebit: WORD = 40B;
    i1, i2, n: CARDINAL;
    b1, b2: STRING;
    IF s1.length # s2.length
      THEN RETURN [FALSE];
    b1 ← s1.base;  i1 ← s1.offset;
    b2 ← s2.base;  i2 ← s2.offset;
    FOR n ← s1.length, n-1 WHILE n > 0
      DO
      IF BITOR[LOOPHOLE[b1[i1]], casebit] # BITOR[LOOPHOLE[b2[i2]], casebit]
        THEN RETURN [FALSE];
      i1 ← i1+1;  i2 ← i2+1;
      ENDLOOP;
    RETURN [TRUE]
    END;

  DeleteSubString: PROCEDURE [s: SubString] =
    BEGIN
    b: STRING = s.base;
    i: CARDINAL ← s.offset;
    j: CARDINAL ← i + s.length;
    WHILE j < b.length
      DO
      b[i] ← b[j];
      i ← i+1;  j ← j+1;
      ENDLOOP;
    b.length ← i;
    RETURN
    END;


  -- routines for bcpl strings

  WordsforBcplString: PROCEDURE[n:INTEGER] RETURNS[INTEGER] =
    BEGIN RETURN[bcplStringHeaderSize+n/CharsPerWord] END;

  bcplStringOverflow: SIGNAL = CODE;

  MesaToBcplString: PROCEDURE[s:STRING, t:POINTER TO bcplSTRING] =
    BEGIN
    w: bytepair;
```

```
    i: CARDINAL;
    j: CARDINAL ← 0;
    IF s.length>0 THEN
      BEGIN
      t.body ← s[0]; j ← j+1;
      FOR i IN [1..SIZE[bcplSTRING]) WHILE j<s.length DO
        w ← bytepair[s[j],0C]; j ← j+1;
        IF j<s.length THEN
          BEGIN w.right ← s[j];  j ← j+1 END;
        t.rest[i] ← w;
        ENDLOOP;
      IF j<s.length THEN SIGNAL bcplStringOverflow;
      END
    ELSE t.body ← 0C;
    t.length ← j;
    END;

  mesaStringOverflow: SIGNAL = CODE;

  BcplToMesaString: PROCEDURE[t:POINTER TO bcplSTRING, s:STRING] =
    BEGIN
    i,j: CARDINAL;
    IF (s.length ← t.length)>s.maxlength THEN
      BEGIN
      SIGNAL mesaStringOverflow;
      s.length ← s.maxlength;
      END;
    IF s.length>0 THEN
      BEGIN
      s[0] ← t.body; i ← 1;
      FOR j IN [1..s.length) DO
        IF j MOD 2 = 0 THEN
          BEGIN
          s[j] ← t.rest[i].right;
          i ← i+1;
          END
        ELSE s[j] ← t.rest[i].left;
        ENDLOOP;
      END;
    END;


  Overflow: SIGNAL = CODE;
  InvalidNumber: SIGNAL = CODE;
  MaxInteger: PRIVATE INTEGER = 32767;
  NUL: CHARACTER = 0C;
  CharZero: CARDINAL = LOOPHOLE['0];
  Space: CHARACTER = 40C;

  StringToNumber: PUBLIC PROCEDURE [s: STRING, radix: CARDINAL]
    RETURNS [v:UNSPECIFIED] =
    BEGIN OPEN InlineDefs;
    char: CHARACTER;
    cp: CARDINAL ← 0;
    v8, v10: CARDINAL ← 0;
    neg: BOOLEAN ← FALSE;
    getchar: PROCEDURE =
      BEGIN
      char ← s[cp];
      IF (cp ← cp+1) > s.length THEN char ← NUL;
      END;

    getchar[];
    WHILE char <= Space DO
      IF char = NUL THEN SIGNAL InvalidNumber;
      getchar[];
      ENDLOOP;
    IF char = '- THEN
      BEGIN neg ← TRUE; getchar[] END;
    WHILE char IN ['0..'9] DO
      v10 ← v10*10 + (LOOPHOLE[char, CARDINAL]-CharZero);
      v8 ← v8*8 + (LOOPHOLE[char, CARDINAL]-CharZero);
      getchar[];
      ENDLOOP;

    BEGIN
```

```
    SELECT LOOPHOLE[BITAND[LOOPHOLE[char],137B],CHARACTER] FROM
      NUL => GOTO noexponent;
      'B => BEGIN v ← v8; radix ← 8; END;
      'D => BEGIN v ← v10; radix ← 10; END;
      ENDCASE => SIGNAL InvalidNumber;
    getchar[]; v10 ← 0;
    WHILE char IN ['0..'9] DO
      v10 ← v10*10 + (LOOPHOLE[char, CARDINAL]-CharZero);
      getchar[];
      ENDLOOP;
    THROUGH [1 .. v10] DO  v ← v*radix  ENDLOOP;
    EXITS noexponent => v ← IF radix = 8 THEN v8 ELSE v10;
    END;

    IF char # NUL THEN SIGNAL InvalidNumber;
    IF neg THEN RETURN[-v];
    END;

  StringToDecimal: PROCEDURE [s: STRING] RETURNS [INTEGER] =
    BEGIN
    RETURN[StringToNumber[s,10]]
    END;

  StringToOctal: PROCEDURE [s: STRING] RETURNS [UNSPECIFIED] =
    BEGIN
    RETURN[StringToNumber[s,8]];
    END;

  END.
```