```
-- File: Strings.Mesa  Edited by Sandman on May 12, 1978  3:13 PM

DIRECTORY
  InlineDefs: FROM "inlinedefs" USING [BITAND, BITOR, DIVMOD],
  Mopcodes: FROM "mopcodes" USING [zKFCB, zPOP],
  SDDefs: FROM "sddefs" USING [sLongDivMod],
  StringDefs: FROM "stringdefs" USING [
    bcplMaxLength, bcplSTRING, bcplStringHeaderSize, CharsPerWord,
    StringHeaderSize, SubString];

DEFINITIONS FROM StringDefs;

Strings: PROGRAM EXPORTS StringDefs SHARES StringDefs = PUBLIC
  BEGIN

  WordsForString: PROCEDURE [nchars: CARDINAL] RETURNS [CARDINAL] =
    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, EqualStrings: 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, EquivalentStrings: 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;

  EqualSubString, 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;

  EquivalentSubString, 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 [nchars:CARDINAL] RETURNS [CARDINAL] =
    BEGIN RETURN[bcplStringHeaderSize+nchars/CharsPerWord] END;

  bcplStringOverflow: SIGNAL = CODE;

  MesaToBcplString: PROCEDURE[s:STRING, t:POINTER TO bcplSTRING] =
    BEGIN
    i: CARDINAL;
    FOR i IN [0..(t.length ← MIN[s.length,bcplMaxLength])) DO
      t.char[i] ← s[i];
```

```
    ENDLOOP;
  IF s.length > bcplMaxLength THEN SIGNAL bcplStringOverflow;
  END;

mesaStringOverflow: SIGNAL = CODE;

BcplToMesaString: PROCEDURE[t:POINTER TO bcplSTRING, s:STRING] =
  BEGIN
  i: CARDINAL;
  FOR i IN [0..(s.length ← MIN[t.length,s.maxlength])) DO
    s[i] ← t.char[i];
    ENDLOOP;
  IF t.length > s.maxlength THEN SIGNAL mesaStringOverflow;
  END;


Overflow: SIGNAL = CODE;
InvalidNumber: SIGNAL = CODE;
NUL: CHARACTER = 0C;
Space: CHARACTER = ' ;

StringToNumber: 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 + (char-'0);
    v8 ← v8*8 + (char-'0);
    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 => GOTO noexponent;
  getchar[]; v10 ← 0;
  WHILE char IN ['0..'9] DO
    v10 ← v10*10 + char-'0;
    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;
```

```
AppendNumber: PROCEDURE [s: STRING, n: CARDINAL, radix: CARDINAL] =
  BEGIN
  xn: PROCEDURE [n: CARDINAL] =
    BEGIN
    r: CARDINAL;
    [n,r] ← InlineDefs.DIVMOD[n,radix];
    IF n # 0 THEN xn[n];
    IF r>9 THEN r ← r + 'A-'0-10;
    AppendChar[s, r+'0];
    END;
  xn[n];
  END;

AppendDecimal: PROCEDURE [s: STRING, n: INTEGER] =
  BEGIN
  IF n<0 THEN
    BEGIN AppendChar[s,'-]; n ← -n END;
  AppendNumber[s,n,10];
  END;

AppendOctal: PROCEDURE [s: STRING, n: UNSPECIFIED] =
  BEGIN
  AppendNumber[s,n,8];
  AppendChar[s,'B];
  END;

AppendLongNumber: PROCEDURE [s: STRING, n: LONG INTEGER, radix: CARDINAL] =
  BEGIN OPEN Mopcodes;
  DivMod: PROCEDURE [n,d: LONG INTEGER]
    RETURNS [q: LONG INTEGER, r: INTEGER] =
    MACHINE CODE
    BEGIN zKFCB, SDDefs.sLongDivMod; zPOP END;
  xn: PROCEDURE [n: LONG INTEGER] =
    BEGIN
    r: INTEGER;
    [n,r] ← DivMod[n,radix];
    IF n # 0 THEN xn[n];
    IF r>9 THEN r ← r + 'A-'0-10;
    AppendChar[s, r+'0];
    END;
  IF n < 0 THEN BEGIN AppendChar[s,'-]; n ← -n END;
  xn[n];
  END;

StringToLongNumber: PROCEDURE [s: STRING, radix: CARDINAL]
  RETURNS [v: LONG INTEGER] =
  BEGIN OPEN InlineDefs;
  char: CHARACTER;
  cp: CARDINAL ← 0;
  exp: CARDINAL;
  v8, v10:  LONG INTEGER ← 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 + CARDINAL[char-'0];
    v8 ← v8*8 + CARDINAL[char-'0];
    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 => GOTO noexponent;
    getchar[]; exp ← 0;
    WHILE char IN ['0..'9] DO
      exp ← exp*10 + char-'0;
      getchar[];
      ENDLOOP;
    THROUGH [1 .. exp] 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;

  END.
```