```
-- file DIType.Mesa
-- last modified by
--                  Sandman, May 5, 1978  8:21 AM
--                  Barbara, June 23, 1978  12:35 PM

DIRECTORY
  DebuggerDefs: FROM "debuggerdefs" USING [FieldContext],
  DIDefs: FROM "didefs" USING [ESPointer, predefinedType, thereESPointer],
  DITypeDefs: FROM "ditypedefs" USING [
    SeiBoolean, SeiCardinal, SeiCharacter, SeiInteger, SeiString,
    SeiUnspecified, SeiLongInteger],
  StringDefs: FROM "stringdefs" USING [SubStringDescriptor],
  SymbolTableDefs: FROM "symboltabledefs" USING [SymbolTableBase],
  SymDefs: FROM "symdefs" USING [
    codeANY, codeBOOLEAN, codeCHARACTER, codeINTEGER, CSEIndex, CTXIndex,
    SEIndex, SENull, typeANY],
  TypePackDefs: FROM "typepackdefs" USING [
    AssignableTypes, EquivalentTypes, TypeHandle];

DIType: PROGRAM
  IMPORTS TypePackDefs, DebuggerDefs
  EXPORTS DITypeDefs = PUBLIC
  BEGIN OPEN DITypeDefs;

  thereESPointer: TYPE = DIDefs.thereESPointer;
  ESPointer: TYPE = DIDefs.ESPointer;

 -- type manipulation

  TypeArray: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN FALSE
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        array => TRUE,
        ENDCASE => FALSE];
    END;

  TypeArrayDesc: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN esp.desc ELSE esp.desc OR
      (WITH s.seb+StrippedType[esp] SELECT FROM
        arraydesc => TRUE,
        ENDCASE => FALSE)];
    END;

  TypeBoolean: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN esp.tsei = SeiBoolean
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        basic => code = SymDefs.codeBOOLEAN,
        ENDCASE => FALSE];
    END;

  TypeCharacter: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN esp.tsei = SeiCharacter
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        basic => code = SymDefs.codeCHARACTER,
        ENDCASE => FALSE];
    END;

  TypeInteger: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL
        THEN (esp.tsei = SeiInteger OR esp.tsei = SeiLongInteger)
      ELSE WITH s.seb+StrippedType[esp] SELECT FROM
        basic => code = SymDefs.codeINTEGER,
        ENDCASE => FALSE];
    END;

  TypeLong: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN esp.tsei = SeiLongInteger
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        long => TRUE,
        ENDCASE => FALSE];
```

```
    END;

  TypePointer: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    IF esp.stbase = NIL THEN RETURN[esp.indirection # 0];
    RETURN[IF esp.stbase = NIL THEN esp.indirection # 0
      ELSE esp.indirection # 0 OR
      (WITH s.seb+StrippedType[esp] SELECT FROM
         pointer => TRUE,
         ENDCASE => FALSE)];
    END;

  TypeProcedure: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN FALSE
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        transfer => mode = procedure,
        ENDCASE => FALSE];
    END;

  TypeRecord: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN FALSE
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        record => TRUE,
        ENDCASE => FALSE];
    END;

  StringContext: SymDefs.CTXIndex = LOOPHOLE[6];

  TypeString: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN esp.tsei = SeiString
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        pointer => s.TypeForm[pointedtotype] = record AND
          DebuggerDefs.FieldContext[esp.stbase, pointedtotype] = StringContext,
        long => WITH s.seb+s.UnderType[rangetype] SELECT FROM
           pointer => s.TypeForm[pointedtotype] = record AND
          DebuggerDefs.FieldContext[esp.stbase, pointedtotype] = StringContext,
           ENDCASE => FALSE,
        ENDCASE => FALSE];
    END;

  TypeUnspec: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN OPEN s: esp.stbase;
    RETURN[IF esp.stbase = NIL THEN esp.tsei = SeiUnspecified
      ELSE WITH s.seb+s.UnderType[esp.tsei] SELECT FROM
        basic => code = SymDefs.codeANY,
        ENDCASE => FALSE];
    END;

  TypeIU: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN
    RETURN[TypeInteger[esp] OR TypeUnspec[esp]];
    END;

  TypeIUP: PROCEDURE [esp: ESPointer] RETURNS [BOOLEAN] =
    BEGIN
    RETURN[TypeInteger[esp] OR TypeUnspec[esp] OR TypePointer[esp]];
    END;

  StrippedType: PROCEDURE [esp: ESPointer] RETURNS [type: SymDefs.CSEIndex] =
    BEGIN OPEN s: esp.stbase;
    type ← s.UnderType[esp.tsei];
    DO
      WITH s.seb+type SELECT FROM
        subrange => type ← s.UnderType[rangetype];
        long => type ← s.UnderType[rangetype];
        ENDCASE => EXIT;
      ENDLOOP;
    RETURN
    END;

  AssignableTypes: PROCEDURE [lhs, rhs: ESPointer] RETURNS [BOOLEAN] =
    BEGIN
    sei: SymDefs.SEIndex;
```

```
    IF TypeArrayDesc[lhs] AND rhs.desc THEN RETURN[TRUE];
    IF lhs.stbase = NIL THEN
      BEGIN
      IF rhs.stbase = NIL THEN RETURN[
        lhs.tsei = SeiUnspecified OR rhs.tsei = SeiUnspecified OR
        lhs.tsei = rhs.tsei];
      sei ← SeiPType[LOOPHOLE[lhs.tsei], rhs.stbase];
      IF sei = SymDefs.SENull THEN RETURN[FALSE];
      lhs.stbase ← rhs.stbase;
      lhs.tsei ← sei;
      END;
    IF rhs.stbase = NIL THEN
      BEGIN
      sei ← SeiPType[LOOPHOLE[rhs.tsei], lhs.stbase];
      IF sei = SymDefs.SENull THEN RETURN[FALSE];
      rhs.stbase ← lhs.stbase;
      rhs.tsei ← sei;
      END;
    RETURN[CheckTypes[lhs, rhs, TypePackDefs.AssignableTypes]];
    END;

  EquivalentTypes: PROCEDURE [esp1, esp2: ESPointer] RETURNS [BOOLEAN] =
    BEGIN
    sei: SymDefs.SEIndex;
    IF TypeArrayDesc[esp1] AND esp2.desc THEN RETURN[TRUE];
    IF esp1.stbase = NIL THEN
      BEGIN
      IF esp2.stbase = NIL THEN RETURN[
        esp1.tsei = SeiUnspecified OR esp2.tsei = SeiUnspecified OR
        esp1.tsei = esp2.tsei];
      sei ← SeiPType[LOOPHOLE[esp1.tsei], esp2.stbase];
      IF sei = SymDefs.SENull THEN RETURN[FALSE];
      esp1.stbase ← esp2.stbase;
      esp1.tsei ← sei;
      END;
    IF esp2.stbase = NIL THEN
      BEGIN
      sei ← SeiPType[LOOPHOLE[esp2.tsei], esp1.stbase];
      IF sei = SymDefs.SENull THEN RETURN[FALSE];
      esp2.stbase ← esp1.stbase;
      esp2.tsei ← sei;
      END;
    RETURN[CheckTypes[esp1, esp2, TypePackDefs.EquivalentTypes]];
    END;

  TypeHandle: TYPE = TypePackDefs.TypeHandle;

  TypeChecker: TYPE = PROCEDURE [TypeHandle, TypeHandle] RETURNS [BOOLEAN];

  CheckTypes: PRIVATE PROCEDURE [lhs, rhs: ESPointer, proc: TypeChecker]
   RETURNS [BOOLEAN] =
    BEGIN
    typeL, typeR: TypeHandle;
    csei: SymDefs.CSEIndex;
    SELECT lhs.indirection FROM
      = rhs.indirection =>
        BEGIN
        IF TypeInteger[lhs] AND TypeInteger[rhs] THEN RETURN[TRUE];
        typeL ← [stb: lhs.stbase, sei: lhs.stbase.UnderType[lhs.tsei]];
        typeR ← [stb: rhs.stbase, sei: rhs.stbase.UnderType[rhs.tsei]];
        RETURN[proc[typeL, typeR]];
        END;
      > rhs.indirection =>
        BEGIN OPEN s: rhs.stbase;
        csei ← s.UnderType[rhs.tsei];
        THROUGH [rhs.indirection..lhs.indirection) DO
          IF csei = SymDefs.typeANY THEN RETURN[TRUE];
          WITH s.seb+csei SELECT FROM
            basic => RETURN[code = SymDefs.codeANY];
            pointer => csei ← s.UnderType[pointedtotype];
            ENDCASE => RETURN[FALSE];
          ENDLOOP;
        typeR ← [stb: rhs.stbase, sei: csei];
        typeL ← [stb: lhs.stbase, sei: lhs.stbase.UnderType[lhs.tsei]];
        RETURN[proc[typeL, typeR]];
        END;
```

```
      < rhs.indirection =>
        BEGIN OPEN s: lhs.stbase;
        csei ← s.UnderType[lhs.tsei];
        THROUGH [lhs.indirection..rhs.indirection) DO
          IF csei = SymDefs.typeANY THEN RETURN[TRUE];
          WITH s.seb+csei SELECT FROM
            basic => RETURN[code = SymDefs.codeANY];
            pointer => csei ← s.UnderType[pointedtotype];
            ENDCASE => RETURN[FALSE];
          ENDLOOP;
        typeL ← [stb: lhs.stbase, sei: csei];
        typeR ← [stb: rhs.stbase, sei: rhs.stbase.UnderType[rhs.tsei]];
        RETURN[proc[typeL, typeR]];
        END;
      ENDCASE;
    END;

  SymbolTableBase: TYPE = SymbolTableDefs.SymbolTableBase;

  SeiPType: PROCEDURE [type: DIDefs.predefinedType, stbase: SymbolTableBase]
    RETURNS [sei: SymDefs.SEIndex] =
    BEGIN
    sei ← IF stbase = NIL THEN (SELECT type FROM
      integer => SeiInteger,
      cardinal => SeiCardinal,
      character => SeiCharacter,
      boolean => SeiBoolean,
      string => SeiString,
      longinteger => SeiLongInteger,
      ENDCASE => SeiUnspecified)
    ELSE FindContextZeroId[stbase, SELECT type FROM
      integer => "INTEGER"L,
      cardinal => "CARDINAL"L,
      character => "CHARACTER"L,
      boolean => "BOOLEAN"L,
      string => "STRING"L,
      longinteger => "LONG INTEGER"L,
      ENDCASE => "UNSPECIFIED"L];
    RETURN
    END;

  BasicContext: SymDefs.CTXIndex = LOOPHOLE[2];

  FindContextZeroId: PROCEDURE [stbase: SymbolTableBase, id: STRING]
    RETURNS [sei: SymDefs.SEIndex] =
    BEGIN OPEN stbase;
    ssd: StringDefs.SubStringDescriptor ←
      [base: id, offset: 0, length: id.length];
    sei ← SearchContext[FindString[@ssd], BasicContext];
    RETURN
    END;

  END...
```