```
**********************************************************
   52059 Thu Oct 11 11:59:43 2018
new/usr/src/tools/ctf/cvt/dwarf.c
9312 ctf: be less clever about skipping 'extern' variables declarations
9864 DWARF->CTF enum conversion needs to be careful of sign
**********************************************************
______unchanged_portion_omitted_

 831 /*
 832  * Most enums (those with members) will be resolved during this first pass.
 833  * Others - those without members (see the file comment) - won't be, and will
 834  * need to wait until the second pass when they can be matched with their full
 835  * definitions.
 836  */
 837 static void
 838 die_enum_create(dwarf_t *dw, Dwarf_Die die, Dwarf_Off off, tdesc_t *tdp)
 839 {
 840         Dwarf_Die mem;
 841         Dwarf_Unsigned uval;
 842         Dwarf_Signed sval;

 844         debug(3, "die %llu: creating enum\n", off);

 846         tdp->t_type = ENUM;

 848         (void) die_unsigned(dw, die, DW_AT_byte_size, &uval, DW_ATTR_REQ);
 849         tdp->t_size = uval;

 851         if ((mem = die_child(dw, die)) != NULL) {
 852                 elist_t **elastp = &tdp->t_emem;

 854                 do {
 855                         elist_t *el;

 857                         if (die_tag(dw, mem) != DW_TAG_enumerator) {
 858                                 /* Nested type declaration */
 859                                 die_create_one(dw, mem);
 860                                 continue;
 861                         }

 863                         el = xcalloc(sizeof (elist_t));
 864                         el->el_name = die_name(dw, mem);

 866                         /*
 867                          * We have to be careful here: newer GCCs generate DWARF
 868                          * where an unsigned value will happily pass
 869                          * die_signed().  Since negative values will fail
 870                          * die_unsigned(), we try that first to make sure we get
 871                          * the right value.
 872                          */
 873                         if (die_unsigned(dw, mem, DW_AT_const_value,
 866                         if (die_signed(dw, mem, DW_AT_const_value, &sval, 0)) {
 867                                 el->el_number = sval;
 868                         } else if (die_unsigned(dw, mem, DW_AT_const_value,
 874                             &uval, 0)) {
 875                                 el->el_number = uval;
 876                         } else if (die_signed(dw, mem, DW_AT_const_value,
 877                             &sval, 0)) {
 878                                 el->el_number = sval;
 879                         } else {
 880                                 terminate("die %llu: enum %llu: member without "
 881                                     "value\n", off, die_off(dw, mem));
 882                         }

 884                         debug(3, "die %llu: enum %llu: created %s = %d\n", off,
 885                             die_off(dw, mem), el->el_name, el->el_number);
```



```
 887                         *elastp = el;
 888                         elastp = &el->el_next;

 890                 } while ((mem = die_sibling(dw, mem)) != NULL);

 892                 hash_add(dw->dw_enumhash, tdp);

 894                 tdp->t_flags |= TDESC_F_RESOLVED;

 896                 if (tdp->t_name != NULL) {
 897                         iidesc_t *ii = xcalloc(sizeof (iidesc_t));
 898                         ii->ii_type = II_SOU;
 899                         ii->ii_name = xstrdup(tdp->t_name);
 900                         ii->ii_dtype = tdp;

 902                         iidesc_add(dw->dw_td->td_iihash, ii);
 903                 }
 904         }
 905 }
______unchanged_portion_omitted_

1645 /*ARGSUSED3*/
1646 static void
1647 die_variable_create(dwarf_t *dw, Dwarf_Die die, Dwarf_Off off, tdesc_t *tdp)
1648 {
1649         iidesc_t *ii;
1650         char *name;

1652         debug(3, "die %llu: creating object definition\n", off);

1654         /* Skip "Non-Defining Declarations" */
1655         if (die_isdecl(dw, die))
1656                 return;
1646         if (die_isdecl(dw, die) || (name = die_name(dw, die)) == NULL)
1647                 return; /* skip prototypes and nameless objects */

1658         /*
1659          * If we find a DIE of "Declarations Completing Non-Defining
1660          * Declarations", we will use the referenced type's DIE.  This isn't
1661          * quite correct, e.g. DW_AT_decl_line will be the forward declaration
1662          * not this site.  It's sufficient for what we need, however: in
1663          * particular, we should find DW_AT_external as needed there.
1664          */
1665         if (die_attr(dw, die, DW_AT_specification, 0) != NULL) {
1666                 Dwarf_Die sdie;
1667                 Dwarf_Off soff;

1669                 soff = die_attr_ref(dw, die, DW_AT_specification);

1671                 if (dwarf_offdie(dw->dw_dw, soff,
1672                     &sdie, &dw->dw_err) != DW_DLV_OK) {
1673                         terminate("dwarf_offdie(%llu) failed: %s\n",
1674                             soff, dwarf_errmsg(dw->dw_err));
1675                 }

1677                 die = sdie;
1678         }

1680         if ((name = die_name(dw, die)) == NULL)
1681                 return;

1683         ii = xcalloc(sizeof (iidesc_t));
1684         ii->ii_type = die_isglobal(dw, die) ? II_GVAR : II_SVAR;
1685         ii->ii_name = name;
1686         ii->ii_dtype = die_lookup_pass1(dw, die, DW_AT_type);
```

```
1687         if (ii->ii_type == II_SVAR)
1688                 ii->ii_owner = xstrdup(dw->dw_cuname);

1690         iidesc_add(dw->dw_td->td_iihash, ii);
1691 }
```

_____***unchanged_portion_omitted_***