```
**********************************************************
   34086 Fri Apr 26 04:01:28 2019
new/usr/src/cmd/ctfdump/ctfdump.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright 2019, Joyent, Inc.
  13  * Copyright (c) 2019, Joyent, Inc.
  14  */

  16 /*
  17  * Dump information about CTF containers.
  18  */

  20 #include <stdio.h>
  21 #include <unistd.h>
  22 #include <libctf.h>
  23 #include <libgen.h>
  24 #include <stdarg.h>
  25 #include <stdlib.h>
  26 #include <stddef.h>
  27 #include <sys/sysmacros.h>
  28 #include <sys/types.h>
  29 #include <sys/stat.h>
  30 #include <sys/note.h>
  31 #include <fcntl.h>
  32 #include <errno.h>
  33 #include <string.h>
  34 #include <strings.h>
  35 #include <err.h>

  37 #define MAX_NAMELEN (512)

  39 typedef enum ctfdump_arg {
  40         CTFDUMP_OBJECTS =       0x001,
  41         CTFDUMP_FUNCTIONS =     0x002,
  42         CTFDUMP_HEADER =        0x004,
  43         CTFDUMP_LABELS =        0x008,
  44         CTFDUMP_STRINGS =       0x010,
  45         CTFDUMP_STATS =         0x020,
  46         CTFDUMP_TYPES =         0x040,
  47         CTFDUMP_DEFAULT =       0x07f,
  48         CTFDUMP_OUTPUT =        0x080,
  49         CTFDUMP_SOURCE =        0x100,
  50 } ctfdump_arg_t;
_____unchanged_portion_omitted_

 677 /*
```



```
 678  * C-style output. This is designed mainly for comparison purposes, and doesn't
 679  * produce directly valid C:
 680  *
 681  * - the declarations are sorted alphabetically not semantically
 682  * - anonymous enums without other users are elided (e.g. IDCS_PROBE_SENT)
 683  * - doubly-pointed-to functions are wrong (e.g. in kiconv_ops_t)
 684  * - anon unions declared within SOUs aren't expanded
 685  * - function arguments aren't expanded recursively
 686  */

 688 static const char *
 688 static void
 689 ctfsrc_refname(ctf_id_t id, char *buf, size_t bufsize)
 690 {
 691         ctf_id_t ref;

 693         if ((ref = ctf_type_reference(g_fp, id)) == CTF_ERR) {
 694                 ctfdump_fatal("failed to get reference type for %ld: "
 695                     "%s\n", id, ctf_errmsg(ctf_errno(g_fp)));
 696         }

 698         return (ctf_type_name(g_fp, ref, buf, bufsize));
 698         (void) ctf_type_name(g_fp, ref, buf, bufsize);
 699 }

 701 static int
 702 ctfsrc_member_cb(const char *member, ctf_id_t type, ulong_t off, void *arg)
 703 {
 704         _NOTE(ARGUNUSED(arg));
 705         char name[MAX_NAMELEN];

 707         if (ctf_type_cname(g_fp, type, name, sizeof (name), member) == NULL) {
 708                 if (ctf_errno(g_fp) != ECTF_NOPARENT) {
 709                         ctfdump_fatal("type %ld missing name: %s\n", type,
 710                             ctf_errmsg(ctf_errno(g_fp)));
 711                 }

 713                 (void) snprintf(name, sizeof (name), "unknown_t %s", member);
 714         }

 716         /*
 717          * A byte offset is friendlier, but we'll print bits too if it's not
 718          * aligned (i.e. a bitfield).
 719          */
 720         if (off % NBBY != 0) {
 721                 printf("\t%s; /* offset: %lu bytes (%lu bits) */\n",
 721                 (void) printf("\t%s; /* offset: %lu bytes (%lu bits) */\n",
 722                     name, off / NBBY, off);
 723         } else {
 724                 printf("\t%s; /* offset: %lu bytes */\n",
 724                 (void) printf("\t%s; /* offset: %lu bytes */\n",
 725                     name, off / NBBY);
 726         }
 727         return (0);
 728 }

 730 static int
 731 ctfsrc_enum_cb(const char *name, int value, void *arg)
 732 {
 733         _NOTE(ARGUNUSED(arg));
 734         printf("\t%s = %d,\n", name, value);
 734         (void) printf("\t%s = %d,\n", name, value);
 735         return (0);
 736 }
_____unchanged_portion_omitted_
```

```
static void
ctfsrc_type(ctf_id_t id, const char *name)
{
        char refname[MAX_NAMELEN] = "unknown_t";
        char refname[MAX_NAMELEN];
        ctf_id_t ref;
        ssize_t size;
        int kind;

        if ((kind = ctf_type_kind(g_fp, id)) == CTF_ERR) {
                ctfdump_fatal("encountered malformed ctf, type %s does not "
                    "have a kind: %s\n", name, ctf_errmsg(ctf_errno(g_fp)));
        }

        switch (kind) {
        case CTF_K_STRUCT:
        case CTF_K_UNION:
                /*
                 * Delay printing anonymous SOUs; a later typedef will usually
                 * pick them up.
                 */
                if (is_anon_refname(name))
                        break;

                if ((size = ctf_type_size(g_fp, id)) == CTF_ERR) {
                        ctfdump_fatal("failed to get size of %s: %s\n", name,
                            ctf_errmsg(ctf_errno(g_fp)));
                }

                printf("%s { /* 0x%x bytes */\n", name, size);
                (void) printf("%s { /* 0x%x bytes */\n", name, size);

                if (ctf_member_iter(g_fp, id, ctfsrc_member_cb, NULL) != 0) {
                        ctfdump_fatal("failed to iterate members of %s: %s\n",
                            name, ctf_errmsg(ctf_errno(g_fp)));
                }

                printf("};\n\n");
                (void) printf("};\n\n");
                break;
        case CTF_K_ENUM:
                /*
                 * This will throw away any anon enum that isn't followed by a
                 * typedef...
                 */
                if (is_anon_refname(name))
                        break;

                printf("%s {\n", name);
                (void) printf("%s {\n", name);

                if (ctf_enum_iter(g_fp, id, ctfsrc_enum_cb, NULL) != 0) {
                        ctfdump_fatal("failed to iterate enumerators of %s: "
                            "%s\n", name, ctf_errmsg(ctf_errno(g_fp)));
                }

                printf("};\n\n");
                (void) printf("};\n\n");
                break;
        case CTF_K_TYPEDEF:
                /*
                 * If this fails, it's probably because the referent type is in
                 * a parent container that was not supplied via -p.
                 */
                if (ctfsrc_refname(id, refname, sizeof (refname)) == NULL) {
                        printf("typedef %s %s;\n\n", refname, name);
```



```
                        break;
                }
                ctfsrc_refname(id, refname, sizeof (refname));

                if (!is_anon_refname(refname)) {
                        (void) ctf_type_cname(g_fp,
                            ctf_type_reference(g_fp, id), refname,
                            sizeof (refname), name);

                        printf("typedef %s;\n\n", refname);
                        (void) printf("typedef %s;\n\n", refname);
                        break;
                }

                ref = ctf_type_reference(g_fp, id);

                if (ctf_type_kind(g_fp, ref) == CTF_K_ENUM) {
                        printf("typedef enum {\n");
                        (void) printf("typedef enum {\n");

                        if (ctf_enum_iter(g_fp, ref,
                            ctfsrc_enum_cb, NULL) != 0) {
                                ctfdump_fatal("failed to iterate enumerators "
                                    "of %s: %s\n", refname,
                                    ctf_errmsg(ctf_errno(g_fp)));
                        }

                        printf("} %s;\n\n", name);
                        (void) printf("} %s;\n\n", name);
                } else {
                        if ((size = ctf_type_size(g_fp, ref)) == CTF_ERR) {
                                ctfdump_fatal("failed to get size of %s: %s\n",
                                    refname, ctf_errmsg(ctf_errno(g_fp)));
                        }

                        printf("typedef %s{ /* 0x%zx bytes */\n",
                        (void) printf("typedef %s{ /* 0x%zx bytes */\n",
                            refname, size);

                        if (ctf_member_iter(g_fp, ref,
                            ctfsrc_member_cb, NULL) != 0) {
                                ctfdump_fatal("failed to iterate members "
                                    "of %s: %s\n", refname,
                                    ctf_errmsg(ctf_errno(g_fp)));
                        }

                        printf("} %s;\n\n", name);
                        (void) printf("} %s;\n\n", name);
                }

                break;
        case CTF_K_FORWARD:
                printf("%s;\n\n", name);
                (void) printf("%s;\n\n", name);
                break;
        case CTF_K_UNKNOWN:
        case CTF_K_INTEGER:
        case CTF_K_FLOAT:
        case CTF_K_POINTER:
        case CTF_K_ARRAY:
        case CTF_K_FUNCTION:
        case CTF_K_VOLATILE:
        case CTF_K_CONST:
        case CTF_K_RESTRICT:
                break;
        default:
```

```
876                 ctfdump_fatal("encountered unknown kind for type %s: %d\n",
877                     name, kind);
878                 break;
879         }
880 }
```

_____*unchanged_portion_omitted_*

```
900 static void
901 ctfsrc_object(ctf_id_t id, const char *name)
902 {
903         char tname[MAX_NAMELEN];

905         if (ctf_type_cname(g_fp, id, tname, sizeof (tname), name) == NULL) {
906                 if (ctf_errno(g_fp) != ECTF_NOPARENT) {
907                         ctfdump_fatal("type %ld missing name: %s\n", id,
908                             ctf_errmsg(ctf_errno(g_fp)));
909                 }
910                 (void) snprintf(tname, sizeof (tname), "unknown_t %s", name);
911         }

913         printf("extern %s;\n", tname);
906         (void) printf("extern %s;\n", tname);
914 }
```

_____*unchanged_portion_omitted_*

```
931 static void
932 ctfsrc_function(ctf_idname_t *idn)
933 {
934         ctf_funcinfo_t *cfi = &idn->ci_funcinfo;
935         char name[MAX_NAMELEN] = "unknown_t";

937         (void) ctf_type_name(g_fp, cfi->ctc_return, name, sizeof (name));

939         printf("extern %s %s(", name, idn->ci_name);
932         (void) printf("extern %s %s(", name, idn->ci_name);

941         if (cfi->ctc_argc != 0) {
942                 ctfdump_fargs_grow(cfi->ctc_argc);
943                 if (ctf_func_args(g_fp, idn->ci_symidx,
944                     g_nfargc, g_fargc) == CTF_ERR) {
945                         ctfdump_fatal("failed to get arguments for function "
946                             "%s: %s\n", idn->ci_name,
947                             ctf_errmsg(ctf_errno(g_fp)));
948                 }

950                 for (size_t i = 0; i < cfi->ctc_argc; i++) {
951                         ctf_id_t aid = g_fargc[i];

953                         (void) strlcpy(name, "unknown_t", sizeof (name));
946                         name[0] = '\0';

955                         (void) ctf_type_name(g_fp, aid, name, sizeof (name));

957                         printf("%s%s", name,
950                         (void) printf("%s%s", name,
958                             i + 1 == cfi->ctc_argc ? "" : ", ");
959                 }
960         } else {
961                 if (!(cfi->ctc_flags & CTF_FUNC_VARARG))
962                         printf("void");
955                         (void) printf("void");
963         }

965         if (cfi->ctc_flags & CTF_FUNC_VARARG)
966                 printf("%s...", cfi->ctc_argc == 0 ? "" : ", ");
959                 (void) printf("%s...", cfi->ctc_argc == 0 ? "" : ", ");
```



```
968         printf(");\n");
961         (void) printf(");\n");
969 }
```

_____*unchanged_portion_omitted_*

```
978 static void
979 ctfdump_source(void)
980 {
981         ulong_t nr_syms = ctf_nr_syms(g_fp);
982         ctf_id_t max_id = ctf_max_id(g_fp);
983         size_t count = 0;

985         printf("/* Types */\n\n");
978         (void) printf("/* Types */\n\n");

987         if ((idnames = calloc(max_id + 1, sizeof (idnames[0]))) == NULL) {
988                 ctfdump_fatal("failed to alloc idnames: %s\n",
989                     strerror(errno));
990         }

992         /*
993          * Prep for any unknown types (most likely, they exist in the parent,
994          * but we weren't given the -p option).
995          */
996         for (size_t i = 0; i <= max_id; i++) {
997                 (void) strlcpy(idnames[i].ci_name, "unknown_t",
998                     sizeof (idnames[i].ci_name));
999         }

1001        if (ctf_type_iter(g_fp, B_TRUE, ctfsrc_collect_types_cb,
1002            idnames) == CTF_ERR) {
1003                warnx("failed to collect types: %s",
1004                    ctf_errmsg(ctf_errno(g_fp)));
1005                g_exit = 1;
1006        }

1008        qsort(idnames, max_id, sizeof (ctf_idname_t), idname_compare);

1010        for (size_t i = 0; i <= max_id; i++) {
 994        for (size_t i = 0; i < max_id; i++) {
1011                if (idnames[i].ci_id != 0)
1012                        ctfsrc_type(idnames[i].ci_id, idnames[i].ci_name);
1013        }

1015        free(idnames);

1017        printf("\n\n/* Data Objects */\n\n");
1001        (void) printf("\n\n/* Data Objects */\n\n");

1019        if ((idnames = calloc(nr_syms, sizeof (idnames[0]))) == NULL) {
1020                ctfdump_fatal("failed to alloc idnames: %s\n",
1021                    strerror(errno));
1022        }

1024        if (ctf_object_iter(g_fp, ctfsrc_collect_objects_cb,
1025            &count) == CTF_ERR) {
1026                warnx("failed to collect objects: %s",
1027                    ctf_errmsg(ctf_errno(g_fp)));
1028                g_exit = 1;
1029        }

1031        qsort(idnames, count, sizeof (ctf_idname_t), idname_compare);

1033        for (size_t i = 0; i < count; i++)
1034                ctfsrc_object(idnames[i].ci_id, idnames[i].ci_name);
```

```
1036         free(idnames);

1038         printf("\n\n/* Functions */\n\n");
1022         (void) printf("\n\n/* Functions */\n\n");

1040         if ((idnames = calloc(nr_syms, sizeof (idnames[0]))) == NULL) {
1041                 ctfdump_fatal("failed to alloc idnames: %s\n",
1042                     strerror(errno));
1043         }

1045         count = 0;

1047         if (ctf_function_iter(g_fp, ctfsrc_collect_functions_cb,
1048             &count) == CTF_ERR) {
1049                 warnx("failed to collect functions: %s",
1050                     ctf_errmsg(ctf_errno(g_fp)));
1051                 g_exit = 1;
1052         }

1054         qsort(idnames, count, sizeof (ctf_idname_t), idname_compare);

1056         for (size_t i = 0; i < count; i++)
1057                 ctfsrc_function(&idnames[i]);

1059         free(idnames);
1060 }
```

*unchanged_portion_omitted_*

```
1096 int
1097 main(int argc, char *argv[])
1098 {
1099         int c, fd, err;
1100         const char *ufile = NULL, *parent = NULL;

1102         g_progname = basename(argv[0]);
1103         while ((c = getopt(argc, argv, ":cdfhlp:sStu:")) != -1) {
1104                 switch (c) {
1105                 case 'c':
1106                         g_dump |= CTFDUMP_SOURCE;
1107                         break;
1108                 case 'd':
1109                         g_dump |= CTFDUMP_OBJECTS;
1110                         break;
1111                 case 'f':
1112                         g_dump |= CTFDUMP_FUNCTIONS;
1113                         break;
1114                 case 'h':
1115                         g_dump |= CTFDUMP_HEADER;
1116                         break;
1117                 case 'l':
1118                         g_dump |= CTFDUMP_LABELS;
1119                         break;
1120                 case 'p':
1121                         parent = optarg;
1122                         break;
1123                 case 's':
1124                         g_dump |= CTFDUMP_STRINGS;
1125                         break;
1126                 case 'S':
1127                         g_dump |= CTFDUMP_STATS;
1128                         break;
1129                 case 't':
1130                         g_dump |= CTFDUMP_TYPES;
1131                         break;
1132                 case 'u':
```



```
1133                         g_dump |= CTFDUMP_OUTPUT;
1134                         ufile = optarg;
1135                         break;
1136                 case '?':
1137                         ctfdump_usage("Unknown option: -%c\n", optopt);
1138                         return (2);
1139                 case ':':
1140                         ctfdump_usage("Option -%c requires an operand\n",
1141                             optopt);
1142                         return (2);
1143                 }
1144         }

1146         argc -= optind;
1147         argv += optind;

1149         if ((g_dump & CTFDUMP_SOURCE) && !!(g_dump & ~CTFDUMP_SOURCE)) {
1150                 ctfdump_usage("-c must be specified on its own\n");
1151                 return (2);
1152         }

1154         /*
1155          * Dump all information except C source by default.
1156          */
1157         if (g_dump == 0)
1158                 g_dump = CTFDUMP_DEFAULT;

1160         if (argc != 1) {
1161                 ctfdump_usage("no file to dump\n");
1162                 return (2);
1163         }

1165         if ((fd = open(argv[0], O_RDONLY)) < 0)
1166                 ctfdump_fatal("failed to open file %s: %s\n", argv[0],
1167                     strerror(errno));

1169         g_fp = ctf_fdopen(fd, &err);
1170         if (g_fp == NULL)
1171                 ctfdump_fatal("failed to open file %s: %s\n", argv[0],
1172                     ctf_errmsg(err));

1174         /*
1175          * Check to see if this file needs a parent. If it does not and we were
1176          * given one, that should be an error. If it does need one and the
1177          * parent is not specified, that is fine, we just won't know how to
1178          * find child types. If we are given a parent, check at least that the
1179          * labels match.
1180          */
1181         if (ctf_parent_name(g_fp) == NULL) {
1182                 if (parent != NULL)
1183                         ctfdump_fatal("cannot use %s as a parent file, %s is "
1184                             "not a child\n", parent, argv[0]);
1185         } else if (parent != NULL) {
1186                 const char *explabel, *label;
1187                 ctf_file_t *pfp = ctf_open(parent, &err);

1189                 if (pfp == NULL)
1190                         ctfdump_fatal("failed to open parent file %s: %s\n",
1191                             parent, ctf_errmsg(err));

1193                 /*
1194                  * Before we import the parent into the child, check that the
1195                  * labels match. While there is also the notion of the parent
1196                  * name, it's less straightforward to match that. Require that
1197                  * labels match.
1198                  */
```

```
1199                 explabel = ctf_parent_label(g_fp);
1200                 label = ctf_label_topmost(pfp);
1201                 if (explabel == NULL || label == NULL ||
1202                     strcmp(explabel, label) != 0) {
1203                         if (label == NULL)
1204                                 label = "<missing>";
1205                         if (explabel == NULL)
1206                                 explabel = "<missing>";
1207                         ctfdump_fatal("label mismatch between parent %s and "
1208                             "child %s, parent has %s, child expects %s\n",
1209                             parent, argv[0], label, explabel);
1210                 }

1212                 if (ctf_import(g_fp, pfp) != 0)
1213                         ctfdump_fatal("failed to import parent %s: %s\n",
1214                             parent, ctf_errmsg(ctf_errno(g_fp)));
1215         } else {
1216                 if (g_dump & CTFDUMP_SOURCE) {
1217                         printf("/* Warning: parent \"%s\" not supplied: many "
1218                             "types will be unknown. */\n\n",
1219                             ctf_parent_name(g_fp));
1220                 } else {
1221                         fprintf(stderr, "warning: parent \"%s\" not supplied: "
1222                             "many types will be unknown\n\n",
1223                             ctf_parent_name(g_fp));
1224                 }
1225         }

1227         if (g_dump & CTFDUMP_SOURCE) {
1228                 ctfdump_source();
1229                 return (0);
1230         }

1232         /*
1233          * If stats is set, we must run through everything exect CTFDUMP_OUTPUT.
1234          * We also do CTFDUMP_STATS last as a result.
1235          */
1236         if (g_dump & CTFDUMP_HEADER)
1237                 ctfdump_header();

1239         if (g_dump & (CTFDUMP_LABELS | CTFDUMP_STATS))
1240                 ctfdump_labels();

1242         if (g_dump & (CTFDUMP_OBJECTS | CTFDUMP_STATS))
1243                 ctfdump_objects();

1245         if (g_dump & (CTFDUMP_FUNCTIONS | CTFDUMP_STATS))
1246                 ctfdump_functions();

1248         if (g_dump & (CTFDUMP_TYPES | CTFDUMP_STATS))
1249                 ctfdump_types();

1251         if (g_dump & (CTFDUMP_STRINGS | CTFDUMP_STATS))
1252                 ctfdump_strings();

1254         if (g_dump & CTFDUMP_STATS)
1255                 ctfdump_stats();

1257         if (g_dump & CTFDUMP_OUTPUT)
1258                 ctfdump_output(ufile);

1260         return (g_exit);
1261 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   12124 Fri Apr 26 04:01:28 2019
new/usr/src/common/ctf/ctf_impl.h
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License, Version 1.0 only
   6  * (the "License").  You may not use this file except in compliance
   7  * with the License.
   8  *
   9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
  10  * or http://www.opensolaris.org/os/licensing.
  11  * See the License for the specific language governing permissions
  12  * and limitations under the License.
  13  *
  14  * When distributing Covered Code, include this CDDL HEADER in each
  15  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  16  * If applicable, add the following below this CDDL HEADER, with the
  17  * fields enclosed by brackets "[]" replaced with your own identifying
  18  * information: Portions Copyright [yyyy] [name of copyright owner]
  19  *
  20  * CDDL HEADER END
  21  */

  23 /*
  24  * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
  25  * Use is subject to license terms.
  26  */
  27 /*
  28  * Copyright 2019, Joyent, Inc.
  28  * Copyright (c) 2015, Joyent, Inc.  All rights reserved.
  29  */

  31 #ifndef _CTF_IMPL_H
  32 #define _CTF_IMPL_H

  34 #include <sys/types.h>
  35 #include <sys/errno.h>
  36 #include <sys/sysmacros.h>
  37 #include <sys/ctf_api.h>

  39 #ifdef _KERNEL

  41 #include <sys/systm.h>
  42 #include <sys/cmn_err.h>
  43 #include <sys/varargs.h>
  44 #include <sys/ddi.h>
  45 #include <sys/sunddi.h>

  47 #define isspace(c) \
  48         ((c) == ' ' || (c) == '\t' || (c) == '\n' || \
  49         (c) == '\r' || (c) == '\f' || (c) == '\v')

  51 #define MAP_FAILED      ((void *)-1)

  53 #else   /* _KERNEL */
```



```
  55 #include <strings.h>
  56 #include <stdlib.h>
  57 #include <stdarg.h>
  58 #include <stdio.h>
  59 #include <limits.h>
  60 #include <ctype.h>
  61 #include <stddef.h>

  63 #endif  /* _KERNEL */

  65 #ifdef  __cplusplus
  66 extern "C" {
  67 #endif

  69 typedef struct ctf_helem {
  70         uint_t h_name;          /* reference to name in string table */
  71         ushort_t h_type;        /* corresponding type ID number */
  72         ushort_t h_next;        /* index of next element in hash chain */
  73 } ctf_helem_t;
_____unchanged_portion_omitted_

 240 #define LCTF_INDEX_TO_TYPEPTR(fp, i) \
 241         ((ctf_type_t *)((uintptr_t)(fp)->ctf_buf + (fp)->ctf_txlate[(i)]))

 243 #define LCTF_INFO_KIND(fp, info)        ((fp)->ctf_fileops->ctfo_get_kind(info))
 244 #define LCTF_INFO_ROOT(fp, info)        ((fp)->ctf_fileops->ctfo_get_root(info))
 245 #define LCTF_INFO_VLEN(fp, info)        ((fp)->ctf_fileops->ctfo_get_vlen(info))

 247 #define LCTF_MMAP       0x0001  /* libctf should munmap buffers on close */
 248 #define LCTF_CHILD      0x0002  /* CTF container is a child */
 249 #define LCTF_RDWR       0x0004  /* CTF container is writable */
 250 #define LCTF_DIRTY      0x0008  /* CTF container has been modified */

 252 #define CTF_ELF_SCN_NAME        ".SUNW_ctf"

 254 extern ssize_t ctf_get_ctt_size(const ctf_file_t *, const ctf_type_t *,
 255     ssize_t *, ssize_t *);

 257 extern const ctf_type_t *ctf_lookup_by_id(ctf_file_t **, ctf_id_t);

 259 extern ctf_file_t *ctf_fdcreate_int(int, int *, ctf_sect_t *);

 261 extern int ctf_hash_create(ctf_hash_t *, ulong_t);
 262 extern int ctf_hash_insert(ctf_hash_t *, ctf_file_t *, ushort_t, uint_t);
 263 extern int ctf_hash_define(ctf_hash_t *, ctf_file_t *, ushort_t, uint_t);
 264 extern ctf_helem_t *ctf_hash_lookup(ctf_hash_t *, ctf_file_t *,
 265     const char *, size_t);
 266 extern uint_t ctf_hash_size(const ctf_hash_t *);
 267 extern void ctf_hash_destroy(ctf_hash_t *);

 269 #define ctf_list_prev(elem)     ((void *)(((ctf_list_t *)(elem))->l_prev))
 270 #define ctf_list_next(elem)     ((void *)(((ctf_list_t *)(elem))->l_next))

 272 extern void ctf_list_append(ctf_list_t *, void *);
 273 extern void ctf_list_prepend(ctf_list_t *, void *);
 274 extern void ctf_list_insert_before(ctf_list_t *, void *, void *);
 275 extern void ctf_list_delete(ctf_list_t *, void *);

 277 extern void ctf_dtd_insert(ctf_file_t *, ctf_dtdef_t *);
 278 extern void ctf_dtd_delete(ctf_file_t *, ctf_dtdef_t *);
 279 extern ctf_dtdef_t *ctf_dtd_lookup(ctf_file_t *, ctf_id_t);

 281 extern void ctf_dsd_delete(ctf_file_t *, ctf_dsdef_t *);
 282 extern void ctf_dld_delete(ctf_file_t *, ctf_dldef_t *);
```

```
 284 extern void ctf_decl_init(ctf_decl_t *, char *, size_t);
 285 extern void ctf_decl_fini(ctf_decl_t *);
 286 extern void ctf_decl_push(ctf_decl_t *, ctf_file_t *, ctf_id_t);
 287 extern void ctf_decl_sprintf(ctf_decl_t *, const char *, ...);

 289 extern const char *ctf_strraw(ctf_file_t *, uint_t);
 290 extern const char *ctf_strptr(ctf_file_t *, uint_t);

 292 extern ctf_file_t *ctf_set_open_errno(int *, int);
 293 extern long ctf_set_errno(ctf_file_t *, int);

 295 extern const void *ctf_sect_mmap(ctf_sect_t *, int);
 296 extern void ctf_sect_munmap(const ctf_sect_t *);

 298 extern void *ctf_data_alloc(size_t);
 299 extern void ctf_data_free(void *, size_t);
 300 extern void ctf_data_protect(void *, size_t);

 302 extern void *ctf_alloc(size_t);
 303 extern void ctf_free(void *, size_t);

 305 extern char *ctf_strdup(const char *);
 306 extern const char *ctf_strerror(int);
 307 extern void ctf_dprintf(const char *, ...);

 309 extern void *ctf_zopen(int *);

 311 extern ctf_id_t ctf_add_encoded(ctf_file_t *, uint_t, const char *,
 312     const ctf_encoding_t *, uint_t);
 313 extern ctf_id_t ctf_add_reftype(ctf_file_t *, uint_t, const char *, ctf_id_t,
 314     uint_t);
 315 extern boolean_t ctf_sym_valid(uintptr_t, int, uint16_t, uint64_t,
 316     uint32_t);

 318 extern const ctf_type_t *ctf_dyn_lookup_by_id(ctf_file_t *, ctf_id_t);
 319 extern int ctf_dyn_array_info(ctf_file_t *, ctf_id_t, ctf_arinfo_t *);

 321 extern const char _CTF_SECTION[];       /* name of CTF ELF section */
 322 extern const char _CTF_NULLSTR[];       /* empty string */

 324 extern int _libctf_version;             /* library client version */
 325 extern int _libctf_debug;               /* debugging messages enabled */

 327 #ifdef  __cplusplus
 328 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   10270 Fri Apr 26 04:01:28 2019
new/usr/src/common/ctf/ctf_lookup.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright 2019, Joyent, Inc.
 */
#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <sys/sysmacros.h>
#include <ctf_impl.h>

/*
 * Compare the given input string and length against a table of known C storage
 * qualifier keywords.  We just ignore these in ctf_lookup_by_name, below.  To
 * do this quickly, we use a pre-computed Perfect Hash Function similar to the
 * technique originally described in the classic paper:
 *
 * R.J. Cichelli, "Minimal Perfect Hash Functions Made Simple",
 * Communications of the ACM, Volume 23, Issue 1, January 1980, pp. 17-19.
 *
 * For an input string S of length N, we use hash H = S[N - 1] + N - 105, which
 * for the current set of qualifiers yields a unique H in the range [0 .. 20].
 * The hash can be modified when the keyword set changes as necessary.  We also
 * store the length of each keyword and check it prior to the final strcmp().
 */
static int
isqualifier(const char *s, size_t len)
{
        static const struct qual {
                const char *q_name;
```



```
                size_t q_len;
        } qhash[] = {
                { "static", 6 }, { "", 0 }, { "", 0 }, { "", 0 },
                { "volatile", 8 }, { "", 0 }, { "", 0 }, { "", 0 }, { "", 0 },
                { "", 0 }, { "auto", 4 }, { "extern", 6 }, { "", 0 }, { "", 0 },
                { "", 0 }, { "", 0 }, { "const", 5 }, { "register", 8 },
                { "", 0 }, { "restrict", 8 }, { "_Restrict", 9 }
        };

        int h = s[len - 1] + (int)len - 105;
        const struct qual *qp = &qhash[h];

        return (h >= 0 && h < sizeof (qhash) / sizeof (qhash[0]) &&
            len == qp->q_len && strncmp(qp->q_name, s, qp->q_len) == 0);
}
```

*unchanged_portion_omitted_*

```
/*
 * Unlike the normal lookup routines, ctf_dyn_*() variants consult both the
 * processed CTF contents of a ctf_file_t as well as the dynamic types in the
 * dtdef list.
 */

const ctf_type_t *
ctf_dyn_lookup_by_id(ctf_file_t *fp, ctf_id_t id)
{
        ctf_file_t **fpp = &fp;
        const ctf_type_t *t;
        ctf_dtdef_t *dtd;

        if ((t = ctf_lookup_by_id(fpp, id)) != NULL)
                return (t);

        if ((dtd = ctf_dtd_lookup(fp, id)) == NULL)
                return (NULL);

        return (&dtd->dtd_data);
}

int
ctf_dyn_array_info(ctf_file_t *infp, ctf_id_t id, ctf_arinfo_t *arinfop)
{
        ctf_file_t *fp = infp;
        const ctf_type_t *t;
        ctf_dtdef_t *dtd;

        if ((t = ctf_lookup_by_id(&fp, id)) != NULL) {

                if (LCTF_INFO_KIND(fp, t->ctt_info) != CTF_K_ARRAY)
                        return (ctf_set_errno(infp, ECTF_NOTARRAY));

                return (ctf_array_info(fp, id, arinfop));
        }

        if ((dtd = ctf_dtd_lookup(fp, id)) == NULL)
                return (ctf_set_errno(infp, ENOENT));

        if (LCTF_INFO_KIND(fp, dtd->dtd_data.ctt_info) != CTF_K_ARRAY)
                return (ctf_set_errno(infp, ECTF_NOTARRAY));

        bcopy(&dtd->dtd_u.dtu_arr, arinfop, sizeof (*arinfop));
        return (0);
}
```

```
**********************************************************
   87377 Fri Apr 26 04:01:29 2019
new/usr/src/lib/libctf/common/ctf_dwarf.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */
  25 /*
  26  * Copyright 2012 Jason King.  All rights reserved.
  27  * Use is subject to license terms.
  28  */

  30 /*
  31  * Copyright 2019, Joyent, Inc.
  31  * Copyright 2019 Joyent, Inc.
  32  */

  34 /*
  35  * CTF DWARF conversion theory.
  36  *
  37  * DWARF data contains a series of compilation units. Each compilation unit
  38  * generally refers to an object file or what once was, in the case of linked
  39  * binaries and shared objects. Each compilation unit has a series of what DWARF
  40  * calls a DIE (Debugging Information Entry). The set of entries that we care
  41  * about have type information stored in a series of attributes. Each DIE also
  42  * has a tag that identifies the kind of attributes that it has.
  43  *
  44  * A given DIE may itself have children. For example, a DIE that represents a
  45  * structure has children which represent members. Whenever we encounter a DIE
  46  * that has children or other values or types associated with it, we recursively
  47  * process those children first so that way we can then refer to the generated
  48  * CTF type id while processing its parent. This reduces the amount of unknowns
  49  * and fixups that we need. It also ensures that we don't accidentally add types
  50  * that an overzealous compiler might add to the DWARF data but aren't used by
  51  * anything in the system.
  52  *
  53  * Once we do a conversion, we store a mapping in an AVL tree that goes from the
```



```
  54  * DWARF's die offset, which is relative to the given compilation unit, to a
  55  * ctf_id_t.
  56  *
  57  * Unfortunately, some compilers actually will emit duplicate entries for a
  58  * given type that look similar, but aren't quite. To that end, we go through
  59  * and do a variant on a merge once we're done processing a single compilation
  60  * unit which deduplicates all of the types that are in the unit.
  61  *
  62  * Finally, if we encounter an object that has multiple compilation units, then
  63  * we'll convert all of the compilation units separately and then do a merge, so
  64  * that way we can result in one single ctf_file_t that represents everything
  65  * for the object.
  66  *
  67  * Conversion Steps
  68  * ----------------
  69  *
  70  * Because a given object we've been given to convert may have multiple
  71  * compilation units, we break the work into two halves. The first half
  72  * processes each compilation unit (potentially in parallel) and then the second
  73  * half optionally merges all of the dies in the first half. First, we'll cover
  74  * what's involved in converting a single ctf_cu_t's dwarf to CTF. This covers
  75  * the work done in ctf_dwarf_convert_one().
  76  *
  77  * An individual ctf_cu_t, which represents a compilation unit, is converted to
  78  * CTF in a series of multiple passes.
  79  *
  80  * Pass 1: During the first pass we walk all of the top-level dies and if we
  81  * find a function, variable, struct, union, enum or typedef, we recursively
  82  * transform all of its types. We don't recurse or process everything, because
  83  * we don't want to add some of the types that compilers may add which are
  84  * effectively unused.
  85  *
  86  * During pass 1, if we encounter any structures or unions we mark them for
  87  * fixing up later. This is necessary because we may not be able to determine
  88  * the full size of a structure at the beginning of time. This will happen if
  89  * the DWARF attribute DW_AT_byte_size is not present for a member. Because of
  90  * this possibility we defer adding members to structures or even converting
  91  * them during pass 1 and save that for pass 2. Adding all of the base
  92  * structures without any of their members helps deal with any circular
  93  * dependencies that we might encounter.
  94  *
  95  * Pass 2: This pass is used to do the first half of fixing up structures and
  96  * unions. Rather than walk the entire type space again, we actually walk the
  97  * list of structures and unions that we marked for later fixing up. Here, we
  98  * iterate over every structure and add members to the underlying ctf_file_t,
  99  * but not to the structs themselves. One might wonder why we don't, and the
 100  * main reason is that libctf requires a ctf_update() be done before adding the
 101  * members to structures or unions.
 102  *
 103  * Pass 3: This pass is used to do the second half of fixing up structures and
 104  * unions. During this part we always go through and add members to structures
 105  * and unions that we added to the container in the previous pass. In addition,
 106  * we set the structure and union's actual size, which may have additional
 107  * padding added by the compiler, it isn't simply the last offset. DWARF always
 108  * guarantees an attribute exists for this. Importantly no ctf_id_t's change
 109  * during pass 2.
 110  *
 111  * Pass 4: The next phase is to add CTF entries for all of the symbols and
 112  * variables that are present in this die. During pass 1 we added entries to a
 113  * map for each variable and function. During this pass, we iterate over the
 114  * symbol table and when we encounter a symbol that we have in our lists of
 115  * translated information which matches, we then add it to the ctf_file_t.
 116  *
 117  * Pass 5: Here we go and look for any weak symbols and functions and see if
 118  * they match anything that we recognize. If so, then we add type information
 119  * for them at this point based on the matching type.
```

```
 120  *
 121  * Pass 6: This pass is actually a variant on a merge. The traditional merge
 122  * process expects there to be no duplicate types. As such, at the end of
 123  * conversion, we do a dedup on all of the types in the system. The
 124  * deduplication process is described in lib/libctf/common/ctf_merge.c.
 125  *
 126  * Once pass 6 is done, we've finished processing the individual compilation
 127  * unit.
 128  *
 129  * The following steps reflect the general process of doing a conversion.
 130  *
 131  * 1) Walk the dwarf section and determine the number of compilation units
 132  * 2) Create a ctf_cu_t for each compilation unit
 133  * 3) Add all ctf_cu_t's to a workq
 134  * 4) Have the workq process each die with ctf_dwarf_convert_one. This itself
 135  *    is comprised of several steps, which were already enumerated.
 136  * 5) If we have multiple cu's, we do a ctf merge of all the dies. The mechanics
 137  *    of the merge are discussed in lib/libctf/common/ctf_merge.c.
 138  * 6) Free everything up and return a ctf_file_t to the user. If we only had a
 139  *    single compilation unit, then we give that to the user. Otherwise, we
 140  *    return the merged ctf_file_t.
 141  *
 142  * Threading
 143  * ---------
 144  *
 145  * The process has been designed to be amenable to threading. Each compilation
 146  * unit has its own type stream, therefore the logical place to divide and
 147  * conquer is at the compilation unit. Each ctf_cu_t has been built to be able
 148  * to be processed independently of the others. It has its own libdwarf handle,
 149  * as a given libdwarf handle may only be used by a single thread at a time.
 150  * This allows the various ctf_cu_t's to be processed in parallel by different
 151  * threads.
 152  *
 153  * All of the ctf_cu_t's are loaded into a workq which allows for a number of
 154  * threads to be specified and used as a thread pool to process all of the
 155  * queued work. We set the number of threads to use in the workq equal to the
 156  * number of threads that the user has specified.
 157  *
 158  * After all of the compilation units have been drained, we use the same number
 159  * of threads when performing a merge of multiple compilation units, if they
 160  * exist.
 161  *
 162  * While all of these different parts do support and allow for multiple threads,
 163  * it's important that when only a single thread is specified, that it be the
 164  * calling thread. This allows the conversion routines to be used in a context
 165  * that doesn't allow additional threads, such as rtld.
 166  *
 167  * Common DWARF Mechanics and Notes
 168  * --------------------------------
 169  *
 170  * At this time, we really only support DWARFv2, though support for DWARFv4 is
 171  * mostly there. There is no intent to support DWARFv3.
 172  *
 173  * Generally types for something are stored in the DW_AT_type attribute. For
 174  * example, a function's return type will be stored in the local DW_AT_type
 175  * attribute while the arguments will be in child DIEs. There are also various
 176  * times when we don't have any DW_AT_type. In that case, the lack of a type
 177  * implies, at least for C, that its C type is void. Because DWARF doesn't emit
 178  * one, we have a synthetic void type that we create and manipulate instead and
 179  * pass it off to consumers on an as-needed basis. If nothing has a void type,
 180  * it will not be emitted.
 181  *
 182  * Architecture Specific Parts
 183  * ---------------------------
 184  *
 185  * The CTF tooling encodes various information about the various architectures
```



```
 186  * in the system. Importantly, the tool assumes that every architecture has a
 187  * data model where long and pointer are the same size. This is currently the
 188  * case, as the two data models illumos supports are ILP32 and LP64.
 189  *
 190  * In addition, we encode the mapping of various floating point sizes to various
 191  * types for each architecture. If a new architecture is being added, it should
 192  * be added to the list. The general design of the ctf conversion tools is to be
 193  * architecture independent. eg. any of the tools here should be able to convert
 194  * any architecture's DWARF into ctf; however, this has not been rigorously
 195  * tested and more importantly, the ctf routines don't currently write out the
 196  * data in an endian-aware form, they only use that of the currently running
 197  * library.
 198  */

 200 #include <libctf_impl.h>
 201 #include <sys/avl.h>
 202 #include <sys/debug.h>
 203 #include <gelf.h>
 204 #include <libdwarf.h>
 205 #include <dwarf.h>
 206 #include <libgen.h>
 207 #include <workq.h>
 208 #include <errno.h>

 210 #define DWARF_VERSION_TWO       2
 211 #define DWARF_VARARGS_NAME      "..."

 213 /*
 214  * Dwarf may refer recursively to other types that we've already processed. To
 215  * see if we've already converted them, we look them up in an AVL tree that's
 216  * sorted by the DWARF id.
 217  */
 218 typedef struct ctf_dwmap {
 219         avl_node_t      cdm_avl;
 220         Dwarf_Off       cdm_off;
 221         Dwarf_Die       cdm_die;
 222         ctf_id_t        cdm_id;
 223         boolean_t       cdm_fix;
 224 } ctf_dwmap_t;
_____unchanged_portion_omitted_

1488 /*
1489  * Given "const int const_array3[11]", GCC7 at least will create a DIE tree of
1490  * DW_TAG_const_type:DW_TAG_array_type:DW_Tag_const_type:<member_type>.
1491  *
1492  * Given C's syntax, this renders out as "const const int const_array3[11]".  To
1493  * get closer to round-tripping (and make the unit tests work), we'll peek for
1494  * this case, and avoid adding the extraneous qualifier if we see that the
1495  * underlying array referent already has the same qualifier.
1496  *
1497  * This is unfortunately less trivial than it could be: this issue applies to
1498  * qualifier sets like "const volatile", as well as multi-dimensional arrays, so
1499  * we need to descend down those.
1500  *
1501  * Returns CTF_ERR on error, or a boolean value otherwise.
1502  */
1503 static int
1504 needed_array_qualifier(ctf_cu_t *cup, int kind, ctf_id_t ref_id)
1505 {
1506         const ctf_type_t *t;
1507         ctf_arinfo_t arinfo;
1508         int akind;

1510         if (kind != CTF_K_CONST && kind != CTF_K_VOLATILE &&
1511             kind != CTF_K_RESTRICT)
1512                 return (1);
```

```
1514         if ((t = ctf_dyn_lookup_by_id(cup->cu_ctfp, ref_id)) == NULL)
1515                 return (CTF_ERR);

1517         if (LCTF_INFO_KIND(cup->cu_ctfp, t->ctt_info) != CTF_K_ARRAY)
1518                 return (1);

1520         if (ctf_dyn_array_info(cup->cu_ctfp, ref_id, &arinfo) != 0)
1521                 return (CTF_ERR);

1523         ctf_id_t id = arinfo.ctr_contents;

1525         for (;;) {
1526                 if ((t = ctf_dyn_lookup_by_id(cup->cu_ctfp, id)) == NULL)
1527                         return (CTF_ERR);

1529                 akind = LCTF_INFO_KIND(cup->cu_ctfp, t->ctt_info);

1531                 if (akind == kind)
1532                         break;

1534                 if (akind == CTF_K_ARRAY) {
1535                         if (ctf_dyn_array_info(cup->cu_ctfp,
1536                             id, &arinfo) != 0)
1537                                 return (CTF_ERR);
1538                         id = arinfo.ctr_contents;
1539                         continue;
1540                 }

1542                 if (akind != CTF_K_CONST && akind != CTF_K_VOLATILE &&
1543                     akind != CTF_K_RESTRICT)
1544                         break;

1546                 id = t->ctt_type;
1547         }

1549         if (kind == akind) {
1550                 ctf_dprintf("ignoring extraneous %s qualifier for array %d\n",
1551                     ctf_kind_name(cup->cu_ctfp, kind), ref_id);
1552         }

1554         return (kind != akind);
1555 }

1557 static int
1558 ctf_dwarf_create_reference(ctf_cu_t *cup, Dwarf_Die die, ctf_id_t *idp,
1559     int kind, int isroot)
1560 {
1561         int ret;
1562         ctf_id_t id;
1563         Dwarf_Die tdie;
1564         char *name;
1565         size_t namelen;

1567         if ((ret = ctf_dwarf_string(cup, die, DW_AT_name, &name)) != 0 &&
1568             ret != ENOENT)
1569                 return (ret);
1570         if (ret == ENOENT) {
1571                 name = NULL;
1572                 namelen = 0;
1573         } else {
1574                 namelen = strlen(name);
1575         }

1577         ctf_dprintf("reference kind %d %s\n", kind, name != NULL ? name : "<>");
```



```
1579         if ((ret = ctf_dwarf_refdie(cup, die, DW_AT_type, &tdie)) != 0) {
1580                 if (ret != ENOENT) {
1581                         ctf_free(name, namelen);
1582                         return (ret);
1583                 }
1584                 if ((id = ctf_dwarf_void(cup)) == CTF_ERR) {
1585                         ctf_free(name, namelen);
1586                         return (ctf_errno(cup->cu_ctfp));
1587                 }
1588         } else {
1589                 if ((ret = ctf_dwarf_convert_type(cup, tdie, &id,
1590                     CTF_ADD_NONROOT)) != 0) {
1591                         ctf_free(name, namelen);
1592                         return (ret);
1593                 }
1594         }

1596         if ((ret = needed_array_qualifier(cup, kind, id)) <= 0) {
1597                 if (ret != 0) {
1598                         ret = (ctf_errno(cup->cu_ctfp));
1599                 } else {
1600                         *idp = id;
1601                 }

1603                 ctf_free(name, namelen);
1604                 return (ret);
1605         }

1607         if ((*idp = ctf_add_reftype(cup->cu_ctfp, isroot, name, id, kind)) ==
1608             CTF_ERR) {
1609                 ctf_free(name, namelen);
1610                 return (ctf_errno(cup->cu_ctfp));
1611         }

1613         ctf_free(name, namelen);
1614         return (ctf_dwmap_add(cup, *idp, die, B_FALSE));
1615 }
_____unchanged_portion_omitted_

1886 static int
1887 ctf_dwarf_function_count(ctf_cu_t *cup, Dwarf_Die die, ctf_funcinfo_t *fip,
1888     boolean_t fptr)
1889 {
1890         int ret;
1891         Dwarf_Die child, sib, arg;

1893         if ((ret = ctf_dwarf_child(cup, die, &child)) != 0)
1894                 return (ret);

1896         arg = child;
1897         while (arg != NULL) {
1898                 Dwarf_Half tag;

1900                 if ((ret = ctf_dwarf_tag(cup, arg, &tag)) != 0)
1901                         return (ret);

1903                 /*
1904                  * We have to check for a varargs type declaration. This will
1824                  * We have to check for a varargs type decleration. This will
1905                  * happen in one of two ways. If we have a function pointer
1906                  * type, then it'll be done with a tag of type
1907                  * DW_TAG_unspecified_parameters. However, it only means we have
1908                  * a variable number of arguments, if we have more than one
1909                  * argument found so far. Otherwise, when we have a function
1910                  * type, it instead uses a formal parameter whose name is '...'
1911                  * to indicate a variable arguments member.
```

```
1912                  *
1913                  * Also, if we have a function pointer, then we have to expect
1914                  * that we might not get a name at all.
1915                  */
1916                 if (tag == DW_TAG_formal_parameter && fptr == B_FALSE) {
1917                         char *name;
1918                         if ((ret = ctf_dwarf_string(cup, die, DW_AT_name,
1919                             &name)) != 0)
1920                                 return (ret);
1921                         if (strcmp(name, DWARF_VARARGS_NAME) == 0)
1922                                 fip->ctc_flags |= CTF_FUNC_VARARG;
1923                         else
1924                                 fip->ctc_argc++;
1925                         ctf_free(name, strlen(name) + 1);
1926                 } else if (tag == DW_TAG_formal_parameter) {
1927                         fip->ctc_argc++;
1928                 } else if (tag == DW_TAG_unspecified_parameters &&
1929                     fip->ctc_argc > 0) {
1930                         fip->ctc_flags |= CTF_FUNC_VARARG;
1931                 }
1932                 if ((ret = ctf_dwarf_sib(cup, arg, &sib)) != 0)
1933                         return (ret);
1934                 arg = sib;
1935         }

1937         return (0);
1938 }
```

*unchanged_portion_omitted_*

```
1986 static int
1987 ctf_dwarf_convert_function(ctf_cu_t *cup, Dwarf_Die die)
1988 {
1909         int ret;
1910         char *name;
1989         ctf_dwfunc_t *cdf;
1990         Dwarf_Die tdie;
1991         Dwarf_Bool b;
1992         char *name;
1993         int ret;

1995         /*
1996          * Functions that don't have a name are generally functions that have
1997          * been inlined and thus most information about them has been lost. If
1998          * we can't get a name, then instead of returning ENOENT, we silently
1999          * swallow the error.
2000          */
2001         if ((ret = ctf_dwarf_string(cup, die, DW_AT_name, &name)) != 0) {
2002                 if (ret == ENOENT)
2003                         return (0);
2004                 return (ret);
2005         }

2007         ctf_dprintf("beginning work on function %s (die %llx)\n",
2008             name, ctf_die_offset(die));

2010         if ((ret = ctf_dwarf_boolean(cup, die, DW_AT_declaration, &b)) != 0) {
2011                 if (ret != ENOENT)
2012                         return (ret);
2013         } else if (b != 0) {
2014                 /*
2015                  * GCC7 at least creates empty DW_AT_declarations for functions
2016                  * defined in headers.  As they lack details on the function
2017                  * prototype, we need to ignore them.  If we later actually
2018                  * see the relevant function's definition, we will see another
2019                  * DW_TAG_subprogram that is more complete.
2020                  */
```



```
2021                 ctf_dprintf("ignoring declaration of function %s (die %llx)\n",
2022                     name, ctf_die_offset(die));
2023                 return (0);
2024         }

1926         ctf_dprintf("beginning work on function %s\n", name);
2026         if ((cdf = ctf_alloc(sizeof (ctf_dwfunc_t))) == NULL) {
2027                 ctf_free(name, strlen(name) + 1);
2028                 return (ENOMEM);
2029         }
2030         bzero(cdf, sizeof (ctf_dwfunc_t));
2031         cdf->cdf_name = name;

2033         if ((ret = ctf_dwarf_refdie(cup, die, DW_AT_type, &tdie)) == 0) {
2034                 if ((ret = ctf_dwarf_convert_type(cup, tdie,
2035                     &(cdf->cdf_fip.ctc_return), CTF_ADD_ROOT)) != 0) {
2036                         ctf_free(name, strlen(name) + 1);
2037                         ctf_free(cdf, sizeof (ctf_dwfunc_t));
2038                         return (ret);
2039                 }
2040         } else if (ret != ENOENT) {
2041                 ctf_free(name, strlen(name) + 1);
2042                 ctf_free(cdf, sizeof (ctf_dwfunc_t));
2043                 return (ret);
2044         } else {
2045                 if ((cdf->cdf_fip.ctc_return = ctf_dwarf_void(cup)) ==
2046                     CTF_ERR) {
2047                         ctf_free(name, strlen(name) + 1);
2048                         ctf_free(cdf, sizeof (ctf_dwfunc_t));
2049                         return (ctf_errno(cup->cu_ctfp));
2050                 }
2051         }

2053         /*
2054          * A function has a number of children, some of which may not be ones we
2055          * care about. Children that we care about have a type of
2056          * DW_TAG_formal_parameter. We're going to do two passes, the first to
2057          * count the arguments, the second to process them. Afterwards, we
2058          * should be good to go ahead and add this function.
2059          *
2060          * Note, we already got the return type by going in and grabbing it out
2061          * of the DW_AT_type.
2062          */
2063         if ((ret = ctf_dwarf_function_count(cup, die, &cdf->cdf_fip,
2064             B_FALSE)) != 0) {
2065                 ctf_free(name, strlen(name) + 1);
2066                 ctf_free(cdf, sizeof (ctf_dwfunc_t));
2067                 return (ret);
2068         }

2070         ctf_dprintf("beginning to convert function arguments %s\n", name);
2071         if (cdf->cdf_fip.ctc_argc != 0) {
2072                 uint_t argc = cdf->cdf_fip.ctc_argc;
2073                 cdf->cdf_argv = ctf_alloc(sizeof (ctf_id_t) * argc);
2074                 if (cdf->cdf_argv == NULL) {
2075                         ctf_free(name, strlen(name) + 1);
2076                         ctf_free(cdf, sizeof (ctf_dwfunc_t));
2077                         return (ENOMEM);
2078                 }
2079                 if ((ret = ctf_dwarf_convert_fargs(cup, die,
2080                     &cdf->cdf_fip, cdf->cdf_argv)) != 0) {
2081                         ctf_free(cdf->cdf_argv, sizeof (ctf_id_t) * argc);
2082                         ctf_free(name, strlen(name) + 1);
2083                         ctf_free(cdf, sizeof (ctf_dwfunc_t));
2084                         return (ret);
2085                 }
```

```
2086         } else {
2087                 cdf->cdf_argv = NULL;
2088         }

2090         if ((ret = ctf_dwarf_isglobal(cup, die, &cdf->cdf_global)) != 0) {
2091                 ctf_free(cdf->cdf_argv, sizeof (ctf_id_t) *
2092                     cdf->cdf_fip.ctc_argc);
2093                 ctf_free(name, strlen(name) + 1);
2094                 ctf_free(cdf, sizeof (ctf_dwfunc_t));
2095                 return (ret);
2096         }

2098         ctf_list_append(&cup->cu_funcs, cdf);
2099         return (ret);
2100 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   24601 Fri Apr 26 04:01:29 2019
new/usr/src/pkg/manifests/system-test-utiltest.mf
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright 2014, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2014 Nexenta Systems, Inc. All rights reserved.
# Copyright 2019, Joyent, Inc.
# Copyright 2017 Joyent, Inc.
# Copyright 2017 Jason King.
#

set name=pkg.fmri value=pkg:/system/test/utiltest@$(PKGVERS)
set name=pkg.description value="Miscellaneous Utility Unit Tests"
set name=pkg.summary value="Utility Unit Test Suite"
set name=info.classification \
    value=org.opensolaris.category.2008:Development/System
set name=variant.arch value=$(ARCH)
dir path=opt/util-tests
dir path=opt/util-tests/bin
dir path=opt/util-tests/runfiles
dir path=opt/util-tests/tests
dir path=opt/util-tests/tests/ctf
dir path=opt/util-tests/tests/ctf/test-merge-dedup
dir path=opt/util-tests/tests/ctf/test-merge-forward
dir path=opt/util-tests/tests/ctf/test-merge-reduction
dir path=opt/util-tests/tests/ctf/test-merge-static
dir path=opt/util-tests/tests/ctf/test-merge-weak
dir path=opt/util-tests/tests/demangle
dir path=opt/util-tests/tests/dis
dir path=opt/util-tests/tests/dis/i386
dir path=opt/util-tests/tests/dis/risc-v
dir path=opt/util-tests/tests/dis/risc-v-c
dir path=opt/util-tests/tests/dis/sparc
dir path=opt/util-tests/tests/files
dir path=opt/util-tests/tests/libnvpair_json
dir path=opt/util-tests/tests/libsff
dir path=opt/util-tests/tests/mergeq
file path=opt/util-tests/README mode=0444
file path=opt/util-tests/bin/print_json mode=0555
file path=opt/util-tests/bin/utiltest mode=0555
file path=opt/util-tests/runfiles/default.run mode=0444
file path=opt/util-tests/tests/allowed-ips mode=0555
file path=opt/util-tests/tests/chown_test mode=0555
file path=opt/util-tests/tests/ctf/Makefile.ctftest.com mode=0555
file path=opt/util-tests/tests/ctf/check-array mode=0555
```



```
file path=opt/util-tests/tests/ctf/check-enum mode=0555
file path=opt/util-tests/tests/ctf/check-float-32 mode=0555
file path=opt/util-tests/tests/ctf/check-float-64 mode=0555
file path=opt/util-tests/tests/ctf/check-forward-32 mode=0555
file path=opt/util-tests/tests/ctf/check-forward-64 mode=0555
file path=opt/util-tests/tests/ctf/check-function mode=0555
file path=opt/util-tests/tests/ctf/check-int-32 mode=0555
file path=opt/util-tests/tests/ctf/check-int-64 mode=0555
file path=opt/util-tests/tests/ctf/check-merge-dedup mode=0555
file path=opt/util-tests/tests/ctf/check-merge-forward-32 mode=0555
file path=opt/util-tests/tests/ctf/check-merge-forward-64 mode=0555
file path=opt/util-tests/tests/ctf/check-merge-reduction mode=0555
file path=opt/util-tests/tests/ctf/check-merge-static mode=0555
file path=opt/util-tests/tests/ctf/check-merge-weak mode=0555
file path=opt/util-tests/tests/ctf/check-qualifiers mode=0555
file path=opt/util-tests/tests/ctf/check-reference mode=0555
file path=opt/util-tests/tests/ctf/check-sou-32 mode=0555
file path=opt/util-tests/tests/ctf/check-sou-64 mode=0555
file path=opt/util-tests/tests/ctf/check-weak mode=0555
file path=opt/util-tests/tests/ctf/ctftest mode=0555
file path=opt/util-tests/tests/ctf/ctftest-convert-no-dwarf mode=0555
file path=opt/util-tests/tests/ctf/ctftest-convert-non-c mode=0555
file path=opt/util-tests/tests/ctf/ctftest-merge-no-ctf mode=0555
file path=opt/util-tests/tests/ctf/precheck mode=0555
file path=opt/util-tests/tests/ctf/test-array.c mode=0555
file path=opt/util-tests/tests/ctf/test-enum.c mode=0555
file path=opt/util-tests/tests/ctf/test-float.c mode=0555
file path=opt/util-tests/tests/ctf/test-forward.c mode=0555
file path=opt/util-tests/tests/ctf/test-function.c mode=0555
file path=opt/util-tests/tests/ctf/test-int.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-dedup/Makefile.ctftest mode=0555
file path=opt/util-tests/tests/ctf/test-merge-dedup/test-merge-1.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-dedup/test-merge-2.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-dedup/test-merge-3.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-dedup/test-merge-dedup.c \
    mode=0555
file path=opt/util-tests/tests/ctf/test-merge-forward/Makefile.ctftest \
    mode=0555
file path=opt/util-tests/tests/ctf/test-merge-forward/test-impl.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-forward/test-merge.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-reduction/Makefile.ctftest \
    mode=0555
file path=opt/util-tests/tests/ctf/test-merge-reduction/mapfile-vers mode=0555
file path=opt/util-tests/tests/ctf/test-merge-reduction/test-global.c \
    mode=0555
file path=opt/util-tests/tests/ctf/test-merge-reduction/test-scoped.c \
    mode=0555
file path=opt/util-tests/tests/ctf/test-merge-static/Makefile.ctftest \
    mode=0555
file path=opt/util-tests/tests/ctf/test-merge-static/test-a.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-static/test-b.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-static/test-c.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-static/test-d.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-static/test-main.c mode=0555
file path=opt/util-tests/tests/ctf/test-merge-weak/Makefile.ctftest mode=0555
file path=opt/util-tests/tests/ctf/test-merge-weak/test-merge-weak.c mode=0555
file path=opt/util-tests/tests/ctf/test-qualifiers.c mode=0555
file path=opt/util-tests/tests/ctf/test-reference.c mode=0555
file path=opt/util-tests/tests/ctf/test-sou.c mode=0555
file path=opt/util-tests/tests/ctf/test-weak.c mode=0555
file path=opt/util-tests/tests/date_test mode=0555
file path=opt/util-tests/tests/demangle/afl-fast mode=0555
file path=opt/util-tests/tests/demangle/gcc-libstdc++ mode=0555
file path=opt/util-tests/tests/demangle/llvm-stdcxxabi mode=0555
file path=opt/util-tests/tests/dis/distest mode=0555
file path=opt/util-tests/tests/dis/i386/32.adx.out mode=0444
```

```
120 file path=opt/util-tests/tests/dis/i386/32.adx.s mode=0444
121 file path=opt/util-tests/tests/dis/i386/32.aes.out mode=0444
122 file path=opt/util-tests/tests/dis/i386/32.aes.s mode=0444
123 file path=opt/util-tests/tests/dis/i386/32.avx.out mode=0444
124 file path=opt/util-tests/tests/dis/i386/32.avx.s mode=0444
125 file path=opt/util-tests/tests/dis/i386/32.avx2-gather.out mode=0444
126 file path=opt/util-tests/tests/dis/i386/32.avx2-gather.s mode=0444
127 file path=opt/util-tests/tests/dis/i386/32.avx2.out mode=0444
128 file path=opt/util-tests/tests/dis/i386/32.avx2.s mode=0444
129 file path=opt/util-tests/tests/dis/i386/32.avx512.out mode=0444
130 file path=opt/util-tests/tests/dis/i386/32.avx512.s mode=0444
131 file path=opt/util-tests/tests/dis/i386/32.bmi1.out mode=0444
132 file path=opt/util-tests/tests/dis/i386/32.bmi1.s mode=0444
133 file path=opt/util-tests/tests/dis/i386/32.bmi2.out mode=0444
134 file path=opt/util-tests/tests/dis/i386/32.bmi2.s mode=0444
135 file path=opt/util-tests/tests/dis/i386/32.clflush.out mode=0444
136 file path=opt/util-tests/tests/dis/i386/32.clflush.s mode=0444
137 file path=opt/util-tests/tests/dis/i386/32.f16c.out mode=0444
138 file path=opt/util-tests/tests/dis/i386/32.f16c.s mode=0444
139 file path=opt/util-tests/tests/dis/i386/32.fma-pd.out mode=0444
140 file path=opt/util-tests/tests/dis/i386/32.fma-pd.s mode=0444
141 file path=opt/util-tests/tests/dis/i386/32.fma-ps.out mode=0444
142 file path=opt/util-tests/tests/dis/i386/32.fma-ps.s mode=0444
143 file path=opt/util-tests/tests/dis/i386/32.fma-sd.out mode=0444
144 file path=opt/util-tests/tests/dis/i386/32.fma-sd.s mode=0444
145 file path=opt/util-tests/tests/dis/i386/32.fma-ss.out mode=0444
146 file path=opt/util-tests/tests/dis/i386/32.fma-ss.s mode=0444
147 file path=opt/util-tests/tests/dis/i386/32.lzcnt.out mode=0444
148 file path=opt/util-tests/tests/dis/i386/32.lzcnt.s mode=0444
149 file path=opt/util-tests/tests/dis/i386/32.movbe.out mode=0444
150 file path=opt/util-tests/tests/dis/i386/32.movbe.s mode=0444
151 file path=opt/util-tests/tests/dis/i386/32.opmask.out mode=0444
152 file path=opt/util-tests/tests/dis/i386/32.opmask.s mode=0444
153 file path=opt/util-tests/tests/dis/i386/32.pclmulqdq.out mode=0444
154 file path=opt/util-tests/tests/dis/i386/32.pclmulqdq.s mode=0444
155 file path=opt/util-tests/tests/dis/i386/32.popcnt.out mode=0444
156 file path=opt/util-tests/tests/dis/i386/32.popcnt.s mode=0444
157 file path=opt/util-tests/tests/dis/i386/32.sha.out mode=0444
158 file path=opt/util-tests/tests/dis/i386/32.sha.s mode=0444
159 file path=opt/util-tests/tests/dis/i386/32.sse-3.out mode=0444
160 file path=opt/util-tests/tests/dis/i386/32.sse-3.s mode=0444
161 file path=opt/util-tests/tests/dis/i386/32.sse-4.1.out mode=0444
162 file path=opt/util-tests/tests/dis/i386/32.sse-4.1.s mode=0444
163 file path=opt/util-tests/tests/dis/i386/32.sse-4.2.out mode=0444
164 file path=opt/util-tests/tests/dis/i386/32.sse-4.2.s mode=0444
165 file path=opt/util-tests/tests/dis/i386/32.ssse3.out mode=0444
166 file path=opt/util-tests/tests/dis/i386/32.ssse3.s mode=0444
167 file path=opt/util-tests/tests/dis/i386/32.xsave.out mode=0444
168 file path=opt/util-tests/tests/dis/i386/32.xsave.s mode=0444
169 file path=opt/util-tests/tests/dis/i386/64.adx.out mode=0444
170 file path=opt/util-tests/tests/dis/i386/64.adx.s mode=0444
171 file path=opt/util-tests/tests/dis/i386/64.aes.out mode=0444
172 file path=opt/util-tests/tests/dis/i386/64.aes.s mode=0444
173 file path=opt/util-tests/tests/dis/i386/64.avx.out mode=0444
174 file path=opt/util-tests/tests/dis/i386/64.avx.s mode=0444
175 file path=opt/util-tests/tests/dis/i386/64.avx2-gather.out mode=0444
176 file path=opt/util-tests/tests/dis/i386/64.avx2-gather.s mode=0444
177 file path=opt/util-tests/tests/dis/i386/64.avx2.out mode=0444
178 file path=opt/util-tests/tests/dis/i386/64.avx2.s mode=0444
179 file path=opt/util-tests/tests/dis/i386/64.avx512.out mode=0444
180 file path=opt/util-tests/tests/dis/i386/64.avx512.s mode=0444
181 file path=opt/util-tests/tests/dis/i386/64.bmi1.out mode=0444
182 file path=opt/util-tests/tests/dis/i386/64.bmi1.s mode=0444
183 file path=opt/util-tests/tests/dis/i386/64.bmi2.out mode=0444
184 file path=opt/util-tests/tests/dis/i386/64.bmi2.s mode=0444
185 file path=opt/util-tests/tests/dis/i386/64.clflush.out mode=0444
```



```
186 file path=opt/util-tests/tests/dis/i386/64.clflush.s mode=0444
187 file path=opt/util-tests/tests/dis/i386/64.ept.out mode=0444
188 file path=opt/util-tests/tests/dis/i386/64.ept.s mode=0444
189 file path=opt/util-tests/tests/dis/i386/64.f16c.out mode=0444
190 file path=opt/util-tests/tests/dis/i386/64.f16c.s mode=0444
191 file path=opt/util-tests/tests/dis/i386/64.fma-pd.out mode=0444
192 file path=opt/util-tests/tests/dis/i386/64.fma-pd.s mode=0444
193 file path=opt/util-tests/tests/dis/i386/64.fma-ps.out mode=0444
194 file path=opt/util-tests/tests/dis/i386/64.fma-ps.s mode=0444
195 file path=opt/util-tests/tests/dis/i386/64.fma-sd.out mode=0444
196 file path=opt/util-tests/tests/dis/i386/64.fma-sd.s mode=0444
197 file path=opt/util-tests/tests/dis/i386/64.fma-ss.out mode=0444
198 file path=opt/util-tests/tests/dis/i386/64.fma-ss.s mode=0444
199 file path=opt/util-tests/tests/dis/i386/64.fsbase.out mode=0444
200 file path=opt/util-tests/tests/dis/i386/64.fsbase.s mode=0444
201 file path=opt/util-tests/tests/dis/i386/64.lzcnt.out mode=0444
202 file path=opt/util-tests/tests/dis/i386/64.lzcnt.s mode=0444
203 file path=opt/util-tests/tests/dis/i386/64.movbe.out mode=0444
204 file path=opt/util-tests/tests/dis/i386/64.movbe.s mode=0444
205 file path=opt/util-tests/tests/dis/i386/64.opmask.out mode=0444
206 file path=opt/util-tests/tests/dis/i386/64.opmask.s mode=0444
207 file path=opt/util-tests/tests/dis/i386/64.pclmulqdq.out mode=0444
208 file path=opt/util-tests/tests/dis/i386/64.pclmulqdq.s mode=0444
209 file path=opt/util-tests/tests/dis/i386/64.popcnt.out mode=0444
210 file path=opt/util-tests/tests/dis/i386/64.popcnt.s mode=0444
211 file path=opt/util-tests/tests/dis/i386/64.random.out mode=0444
212 file path=opt/util-tests/tests/dis/i386/64.random.s mode=0444
213 file path=opt/util-tests/tests/dis/i386/64.sha.out mode=0444
214 file path=opt/util-tests/tests/dis/i386/64.sha.s mode=0444
215 file path=opt/util-tests/tests/dis/i386/64.sse-3.out mode=0444
216 file path=opt/util-tests/tests/dis/i386/64.sse-3.s mode=0444
217 file path=opt/util-tests/tests/dis/i386/64.sse-4.1.out mode=0444
218 file path=opt/util-tests/tests/dis/i386/64.sse-4.1.s mode=0444
219 file path=opt/util-tests/tests/dis/i386/64.sse-4.2.out mode=0444
220 file path=opt/util-tests/tests/dis/i386/64.sse-4.2.s mode=0444
221 file path=opt/util-tests/tests/dis/i386/64.ssse3.out mode=0444
222 file path=opt/util-tests/tests/dis/i386/64.ssse3.s mode=0444
223 file path=opt/util-tests/tests/dis/i386/64.vmx.out mode=0444
224 file path=opt/util-tests/tests/dis/i386/64.vmx.s mode=0444
225 file path=opt/util-tests/tests/dis/i386/64.xsave.out mode=0444
226 file path=opt/util-tests/tests/dis/i386/64.xsave.s mode=0444
227 file path=opt/util-tests/tests/dis/i386/tst.cpuid.out mode=0444
228 file path=opt/util-tests/tests/dis/i386/tst.cpuid.s mode=0444
229 file path=opt/util-tests/tests/dis/i386/tst.fence.out mode=0444
230 file path=opt/util-tests/tests/dis/i386/tst.fence.s mode=0444
231 file path=opt/util-tests/tests/dis/i386/tst.msr.out mode=0444
232 file path=opt/util-tests/tests/dis/i386/tst.msr.s mode=0444
233 file path=opt/util-tests/tests/dis/i386/tst.mwait.out mode=0444
234 file path=opt/util-tests/tests/dis/i386/tst.mwait.s mode=0444
235 file path=opt/util-tests/tests/dis/i386/tst.random.out mode=0444
236 file path=opt/util-tests/tests/dis/i386/tst.random.s mode=0444
237 file path=opt/util-tests/tests/dis/i386/tst.sep.out mode=0444
238 file path=opt/util-tests/tests/dis/i386/tst.sep.s mode=0444
239 file path=opt/util-tests/tests/dis/i386/tst.smap.out mode=0444
240 file path=opt/util-tests/tests/dis/i386/tst.smap.s mode=0444
241 file path=opt/util-tests/tests/dis/i386/tst.tsc.out mode=0444
242 file path=opt/util-tests/tests/dis/i386/tst.tsc.s mode=0444
243 file path=opt/util-tests/tests/dis/i386/tst.vmx.out mode=0444
244 file path=opt/util-tests/tests/dis/i386/tst.vmx.s mode=0444
245 file path=opt/util-tests/tests/dis/risc-v-c/32.ldsr.out mode=0444
246 file path=opt/util-tests/tests/dis/risc-v-c/32.ldsr.s mode=0444
247 file path=opt/util-tests/tests/dis/risc-v-c/64.int.out mode=0444
248 file path=opt/util-tests/tests/dis/risc-v-c/64.int.s mode=0444
249 file path=opt/util-tests/tests/dis/risc-v-c/64.ldsr.out mode=0444
250 file path=opt/util-tests/tests/dis/risc-v-c/64.ldsr.s mode=0444
251 file path=opt/util-tests/tests/dis/risc-v-c/tst.int.out mode=0444
```

```
252 file path=opt/util-tests/tests/dis/risc-v-c/tst.int.s mode=0444
253 file path=opt/util-tests/tests/dis/risc-v-c/tst.ldsr.out mode=0444
254 file path=opt/util-tests/tests/dis/risc-v-c/tst.ldsr.s mode=0444
255 file path=opt/util-tests/tests/dis/risc-v/64.rv64a.out mode=0444
256 file path=opt/util-tests/tests/dis/risc-v/64.rv64a.s mode=0444
257 file path=opt/util-tests/tests/dis/risc-v/64.rv64d.out mode=0444
258 file path=opt/util-tests/tests/dis/risc-v/64.rv64d.s mode=0444
259 file path=opt/util-tests/tests/dis/risc-v/64.rv64f.out mode=0444
260 file path=opt/util-tests/tests/dis/risc-v/64.rv64f.s mode=0444
261 file path=opt/util-tests/tests/dis/risc-v/64.rv64i.out mode=0444
262 file path=opt/util-tests/tests/dis/risc-v/64.rv64i.s mode=0444
263 file path=opt/util-tests/tests/dis/risc-v/64.rv64m.out mode=0444
264 file path=opt/util-tests/tests/dis/risc-v/64.rv64m.s mode=0444
265 file path=opt/util-tests/tests/dis/risc-v/tst.csr.out mode=0444
266 file path=opt/util-tests/tests/dis/risc-v/tst.csr.s mode=0444
267 file path=opt/util-tests/tests/dis/risc-v/tst.fpregs.out mode=0444
268 file path=opt/util-tests/tests/dis/risc-v/tst.fpregs.s mode=0444
269 file path=opt/util-tests/tests/dis/risc-v/tst.pseudo.out mode=0444
270 file path=opt/util-tests/tests/dis/risc-v/tst.pseudo.s mode=0444
271 file path=opt/util-tests/tests/dis/risc-v/tst.regs.out mode=0444
272 file path=opt/util-tests/tests/dis/risc-v/tst.regs.s mode=0444
273 file path=opt/util-tests/tests/dis/risc-v/tst.rv32a.out mode=0444
274 file path=opt/util-tests/tests/dis/risc-v/tst.rv32a.s mode=0444
275 file path=opt/util-tests/tests/dis/risc-v/tst.rv32d.out mode=0444
276 file path=opt/util-tests/tests/dis/risc-v/tst.rv32d.s mode=0444
277 file path=opt/util-tests/tests/dis/risc-v/tst.rv32f.out mode=0444
278 file path=opt/util-tests/tests/dis/risc-v/tst.rv32f.s mode=0444
279 file path=opt/util-tests/tests/dis/risc-v/tst.rv32i.out mode=0444
280 file path=opt/util-tests/tests/dis/risc-v/tst.rv32i.s mode=0444
281 file path=opt/util-tests/tests/dis/risc-v/tst.rv32m.out mode=0444
282 file path=opt/util-tests/tests/dis/risc-v/tst.rv32m.s mode=0444
283 file path=opt/util-tests/tests/dis/risc-v/tst.supervisor.out mode=0444
284 file path=opt/util-tests/tests/dis/risc-v/tst.supervisor.s mode=0444
285 file path=opt/util-tests/tests/dis/sparc/tst.regs.out mode=0444
286 file path=opt/util-tests/tests/dis/sparc/tst.regs.s mode=0444
287 file path=opt/util-tests/tests/files/cout0 mode=0444
288 file path=opt/util-tests/tests/files/cout1 mode=0444
289 file path=opt/util-tests/tests/files/cout10 mode=0444
290 file path=opt/util-tests/tests/files/cout11 mode=0444
291 file path=opt/util-tests/tests/files/cout2 mode=0444
292 file path=opt/util-tests/tests/files/cout3 mode=0444
293 file path=opt/util-tests/tests/files/cout4 mode=0444
294 file path=opt/util-tests/tests/files/cout5 mode=0444
295 file path=opt/util-tests/tests/files/cout6 mode=0444
296 file path=opt/util-tests/tests/files/cout7 mode=0444
297 file path=opt/util-tests/tests/files/cout8 mode=0444
298 file path=opt/util-tests/tests/files/cout9 mode=0444
299 file path=opt/util-tests/tests/files/gout0 mode=0444
300 file path=opt/util-tests/tests/files/gout1 mode=0444
301 file path=opt/util-tests/tests/files/gout10 mode=0444
302 file path=opt/util-tests/tests/files/gout11 mode=0444
303 file path=opt/util-tests/tests/files/gout12 mode=0444
304 file path=opt/util-tests/tests/files/gout13 mode=0444
305 file path=opt/util-tests/tests/files/gout14 mode=0444
306 file path=opt/util-tests/tests/files/gout15 mode=0444
307 file path=opt/util-tests/tests/files/gout16 mode=0444
308 file path=opt/util-tests/tests/files/gout17 mode=0444
309 file path=opt/util-tests/tests/files/gout18 mode=0444
310 file path=opt/util-tests/tests/files/gout19 mode=0444
311 file path=opt/util-tests/tests/files/gout2 mode=0444
312 file path=opt/util-tests/tests/files/gout20 mode=0444
313 file path=opt/util-tests/tests/files/gout21 mode=0444
314 file path=opt/util-tests/tests/files/gout22 mode=0444
315 file path=opt/util-tests/tests/files/gout23 mode=0444
316 file path=opt/util-tests/tests/files/gout24 mode=0444
317 file path=opt/util-tests/tests/files/gout25 mode=0444
```



```
318 file path=opt/util-tests/tests/files/gout26 mode=0444
319 file path=opt/util-tests/tests/files/gout27 mode=0444
320 file path=opt/util-tests/tests/files/gout28 mode=0444
321 file path=opt/util-tests/tests/files/gout29 mode=0444
322 file path=opt/util-tests/tests/files/gout3 mode=0444
323 file path=opt/util-tests/tests/files/gout30 mode=0444
324 file path=opt/util-tests/tests/files/gout31 mode=0444
325 file path=opt/util-tests/tests/files/gout32 mode=0444
326 file path=opt/util-tests/tests/files/gout33 mode=0444
327 file path=opt/util-tests/tests/files/gout34 mode=0444
328 file path=opt/util-tests/tests/files/gout35 mode=0444
329 file path=opt/util-tests/tests/files/gout36 mode=0444
330 file path=opt/util-tests/tests/files/gout37 mode=0444
331 file path=opt/util-tests/tests/files/gout38 mode=0444
332 file path=opt/util-tests/tests/files/gout39 mode=0444
333 file path=opt/util-tests/tests/files/gout4 mode=0444
334 file path=opt/util-tests/tests/files/gout40 mode=0444
335 file path=opt/util-tests/tests/files/gout41 mode=0444
336 file path=opt/util-tests/tests/files/gout42 mode=0444
337 file path=opt/util-tests/tests/files/gout43 mode=0444
338 file path=opt/util-tests/tests/files/gout44 mode=0444
339 file path=opt/util-tests/tests/files/gout45 mode=0444
340 file path=opt/util-tests/tests/files/gout46 mode=0444
341 file path=opt/util-tests/tests/files/gout47 mode=0444
342 file path=opt/util-tests/tests/files/gout48 mode=0444
343 file path=opt/util-tests/tests/files/gout49 mode=0444
344 file path=opt/util-tests/tests/files/gout5 mode=0444
345 file path=opt/util-tests/tests/files/gout50 mode=0444
346 file path=opt/util-tests/tests/files/gout51 mode=0444
347 file path=opt/util-tests/tests/files/gout52 mode=0444
348 file path=opt/util-tests/tests/files/gout53 mode=0444
349 file path=opt/util-tests/tests/files/gout54 mode=0444
350 file path=opt/util-tests/tests/files/gout55 mode=0444
351 file path=opt/util-tests/tests/files/gout56 mode=0444
352 file path=opt/util-tests/tests/files/gout57 mode=0444
353 file path=opt/util-tests/tests/files/gout58 mode=0444
354 file path=opt/util-tests/tests/files/gout59 mode=0444
355 file path=opt/util-tests/tests/files/gout6 mode=0444
356 file path=opt/util-tests/tests/files/gout60 mode=0444
357 file path=opt/util-tests/tests/files/gout61 mode=0444
358 file path=opt/util-tests/tests/files/gout62 mode=0444
359 file path=opt/util-tests/tests/files/gout63 mode=0444
360 file path=opt/util-tests/tests/files/gout64 mode=0444
361 file path=opt/util-tests/tests/files/gout65 mode=0444
362 file path=opt/util-tests/tests/files/gout66 mode=0444
363 file path=opt/util-tests/tests/files/gout7 mode=0444
364 file path=opt/util-tests/tests/files/gout8 mode=0444
365 file path=opt/util-tests/tests/files/gout9 mode=0444
366 file path=opt/util-tests/tests/files/test0 mode=0444
367 file path=opt/util-tests/tests/files/test1 mode=0444
368 file path=opt/util-tests/tests/files/test2 mode=0444
369 file path=opt/util-tests/tests/files/test3 mode=0444
370 file path=opt/util-tests/tests/files/test4 mode=0444
371 file path=opt/util-tests/tests/files/test5 mode=0444
372 file path=opt/util-tests/tests/files/test6 mode=0444
373 file path=opt/util-tests/tests/files/test7 mode=0444
374 file path=opt/util-tests/tests/files/testnl mode=0444
375 file path=opt/util-tests/tests/grep_test mode=0555
376 file path=opt/util-tests/tests/iconv_test mode=0555
377 file path=opt/util-tests/tests/libnvpair_json/json_00_blank mode=0555
378 file path=opt/util-tests/tests/libnvpair_json/json_01_boolean mode=0555
379 file path=opt/util-tests/tests/libnvpair_json/json_02_numbers mode=0555
380 file path=opt/util-tests/tests/libnvpair_json/json_03_empty_arrays mode=0555
381 file path=opt/util-tests/tests/libnvpair_json/json_04_number_arrays mode=0555
382 file path=opt/util-tests/tests/libnvpair_json/json_05_strings mode=0555
383 file path=opt/util-tests/tests/libnvpair_json/json_06_nested mode=0555
```

```
 384 file path=opt/util-tests/tests/libnvpair_json/json_07_nested_arrays mode=0555
 385 file path=opt/util-tests/tests/libnvpair_json/json_common mode=0555
 386 file path=opt/util-tests/tests/libsff/libsff mode=0555
 387 file path=opt/util-tests/tests/libsff/libsff_8472 mode=0555
 388 file path=opt/util-tests/tests/libsff/libsff_8472.out mode=0444
 389 file path=opt/util-tests/tests/libsff/libsff_8636_diag mode=0555
 390 file path=opt/util-tests/tests/libsff/libsff_8636_diag.out mode=0444
 391 file path=opt/util-tests/tests/libsff/libsff_8636_extspec mode=0555
 392 file path=opt/util-tests/tests/libsff/libsff_8636_extspec.out mode=0444
 393 file path=opt/util-tests/tests/libsff/libsff_8636_tech mode=0555
 394 file path=opt/util-tests/tests/libsff/libsff_8636_tech.out mode=0444
 395 file path=opt/util-tests/tests/libsff/libsff_8636_temp mode=0555
 396 file path=opt/util-tests/tests/libsff/libsff_8636_temp.out mode=0444
 397 file path=opt/util-tests/tests/libsff/libsff_br mode=0555
 398 file path=opt/util-tests/tests/libsff/libsff_br.out mode=0444
 399 file path=opt/util-tests/tests/libsff/libsff_compliance mode=0555
 400 file path=opt/util-tests/tests/libsff/libsff_compliance.out mode=0444
 401 file path=opt/util-tests/tests/libsff/libsff_conn mode=0555
 402 file path=opt/util-tests/tests/libsff/libsff_conn.out mode=0444
 403 file path=opt/util-tests/tests/libsff/libsff_efault mode=0555
 404 file path=opt/util-tests/tests/libsff/libsff_einval mode=0555
 405 file path=opt/util-tests/tests/libsff/libsff_enc mode=0555
 406 file path=opt/util-tests/tests/libsff/libsff_enc.out mode=0444
 407 file path=opt/util-tests/tests/libsff/libsff_ident mode=0555
 408 file path=opt/util-tests/tests/libsff/libsff_ident.out mode=0444
 409 file path=opt/util-tests/tests/libsff/libsff_lengths mode=0555
 410 file path=opt/util-tests/tests/libsff/libsff_lengths.out mode=0444
 411 file path=opt/util-tests/tests/libsff/libsff_opts mode=0555
 412 file path=opt/util-tests/tests/libsff/libsff_opts.out mode=0444
 413 file path=opt/util-tests/tests/libsff/libsff_strings mode=0555
 414 file path=opt/util-tests/tests/libsff/libsff_wave mode=0555
 415 file path=opt/util-tests/tests/libsff/libsff_wave.out mode=0444
 416 file path=opt/util-tests/tests/mergeq/mqt mode=0555
 417 file path=opt/util-tests/tests/mergeq/wqt mode=0555
 418 file path=opt/util-tests/tests/printf_test mode=0555
 419 file path=opt/util-tests/tests/set-linkprop mode=0555
 420 file path=opt/util-tests/tests/xargs_test mode=0555
 421 license lic_CDDL license=lic_CDDL
 422 license usr/src/lib/libdemangle/THIRDPARTYLICENSE \
 423     license=usr/src/lib/libdemangle/THIRDPARTYLICENSE
 424 depend fmri=system/library/iconv/utf-8 type=require
 425 depend fmri=system/test/testrunner type=require
```

```
**********************************************************
    3451 Fri Apr 26 04:01:29 2019
new/usr/src/test/util-tests/tests/ctf/Makefile
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2019, Joyent, Inc.
# Copyright (c) 2019, Joyent, Inc.
#

include $(SRC)/Makefile.master

ROOTOPTPKG = $(ROOT)/opt/util-tests
TESTDIR = $(ROOTOPTPKG)/tests/ctf

SCRIPTS =       precheck.ksh \
                ctftest.ksh \
                ctftest-convert-non-c.ksh \
                ctftest-convert-no-dwarf.ksh \
                ctftest-merge-no-ctf.ksh \

TESTS =         test-float.c \
                test-reference.c \
                test-int.c \
                test-array.c \
                test-enum.c \
                test-forward.c \
                test-sou.c \
                test-function.c \
                test-qualifiers.c \
                test-merge-static/Makefile.ctftest \
                test-merge-static/test-a.c \
                test-merge-static/test-b.c \
                test-merge-static/test-c.c \
                test-merge-static/test-d.c \
                test-merge-static/test-main.c \
                test-merge-forward/Makefile.ctftest \
                test-merge-forward/test-impl.c \
                test-merge-forward/test-merge.c \
                test-merge-dedup/Makefile.ctftest \
                test-merge-dedup/test-merge-1.c \
                test-merge-dedup/test-merge-2.c \
                test-merge-dedup/test-merge-3.c \
                test-merge-dedup/test-merge-dedup.c \
                test-merge-reduction/Makefile.ctftest \
                test-merge-reduction/mapfile-vers \
                test-merge-reduction/test-global.c \
                test-merge-reduction/test-scoped.c \
```



```
                test-merge-weak/Makefile.ctftest \
                test-merge-weak/test-merge-weak.c \
                test-weak.c \
                Makefile.ctftest.com

MAKEDIRS =      test-merge-static \
                test-merge-forward \
                test-merge-dedup \
                test-merge-reduction \
                test-merge-weak

CHECKS =        check-float-32 \
                check-float-64 \
                check-int-32 \
                check-int-64 \
                check-reference \
                check-array \
                check-enum \
                check-sou-32 \
                check-sou-64 \
                check-forward-32 \
                check-forward-64 \
                check-function \
                check-qualifiers \
                check-merge-static \
                check-merge-forward-32 \
                check-merge-forward-64 \
                check-merge-dedup \
                check-merge-reduction \
                check-merge-weak \
                check-weak

COMMON_OBJS =   check-common.o
ALL_OBJS =      $(CHECKS:%=%.o) $(CHECKS:%-32=%.32.o) $(CHECKS:%-64=%.64.o) $(CO

ROOTTESTS =     $(TESTS:%=$(TESTDIR)/%)
ROOTMAKEDIRS =  $(MAKEDIRS:%=$(TESTDIR)/%)
ROOTCHECKS =    $(CHECKS:%=$(TESTDIR)/%)
ROOTSCRIPTS =   $(SCRIPTS:%.ksh=$(TESTDIR)/%)

ROOTTESTS       := FILEMODE =   0444
ROOTCHECKS      := FILEMODE =   0555
ROOTSCRIPTS     := FILEMODE =   0555

include $(SRC)/cmd/Makefile.cmd
include $(SRC)/test/Makefile.com

CSTD = $(CSTD_GNU99)

LDLIBS +=       -lctf

check-merge-static :=   LDLIBS += -lelf

all: $(CHECKS)

install: all $(ROOTTESTS) $(ROOTCHECKS) $(ROOTSCRIPTS)

$(CHECKS): $(COMMON_OBJS)

clean:
        $(RM) $(ALL_OBJS)

clobber: clean
        $(RM) $(CHECKS)

$(ROOTTESTS): $(TESTDIR) $(ROOTMAKEDIRS) $(TESTS)
```

```
120 $(ROOTCHECKS): $(TESTDIR) $(CHECKS)
121 $(ROOTSCRIPTS): $(TESTDIR) $(SCRIPTS)

123 $(TESTDIR):
124         $(INS.dir)

126 $(ROOTMAKEDIRS):
127         $(INS.dir)

129 $(TESTDIR)/%: %
130         $(INS.file)

132 $(TESTDIR)/%: %.ksh
133         $(INS.rename)

135 %.o: %.c
136         $(COMPILE.c) -o $@ $<
137         $(POST_PROCESS_O)

139 %.32.o: %.c
140         $(COMPILE.c) -o $@ $<
141         $(POST_PROCESS_O)

143 %.64.o: %.c
144         $(COMPILE.c) -DTARGET_LP64 -o $@ $<
145         $(POST_PROCESS_O)

147 %-32: %.32.o
148         $(LINK.c) -o $@ $< $(COMMON_OBJS) $(LDLIBS)
149         $(POST_PROCESS)

151 %-64: %.64.o
152         $(LINK.c) -o $@ $< $(COMMON_OBJS) $(LDLIBS)
153         $(POST_PROCESS)

155 %: %.o
156         $(LINK.c) -o $@ $< $(COMMON_OBJS) $(LDLIBS)
157         $(POST_PROCESS)
```

```
**********************************************************
    2902 Fri Apr 26 04:01:29 2019
new/usr/src/test/util-tests/tests/ctf/check-array.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright 2019, Joyent, Inc.
  13  * Copyright (c) 2019, Joyent, Inc.
  14  */

  16 /*
  17  * Check that we properly generate basic nested arrays.
  18  */

  20 #include "check-common.h"

  22 static check_number_t check_base[] = {
  23         { "char", CTF_K_INTEGER, CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 },
  24         { "int", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 32 },
  25         { "double", CTF_K_FLOAT, CTF_FP_DOUBLE, 0, 64 },
  26         { NULL }
  27 };
______unchanged_portion_omitted_

  83 int
  84 main(int argc, char *argv[])
  85 {
  86         int i, ret = 0;

  88         if (argc < 2) {
  89                 errx(EXIT_FAILURE, "missing test files");
  90         }

  92         for (i = 1; i < argc; i++) {
  93                 ctf_file_t *fp;
  94                 uint_t d;

  96                 if ((fp = ctf_open(argv[i], &ret)) == NULL) {
  97                         warnx("failed to open %s: %s", argv[i],
  98                             ctf_errmsg(ret));
  99                         ret = EXIT_FAILURE;
 100                         continue;
 101                 }
 102                 if (!ctftest_check_numbers(fp, check_base))
 103                         ret = EXIT_FAILURE;
 104                 if (!ctftest_check_symbols(fp, check_syms))
 105                         ret = EXIT_FAILURE;
 106                 for (d = 0; descents[d].cdt_sym != NULL; d++) {
```



```
 107                         if (!ctftest_check_descent(descents[d].cdt_sym, fp,
 108                             descents[d].cdt_tests, B_FALSE)) {
 108                             descents[d].cdt_tests)) {
 109                                 ret = EXIT_FAILURE;
 110                         }
 111                 }
 112                 ctf_close(fp);
 113         }

 115         return (ret);
 116 }
______unchanged_portion_omitted_
```

```
**********************************************************
   18876 Fri Apr 26 04:01:30 2019
new/usr/src/test/util-tests/tests/ctf/check-common.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2019, Joyent, Inc.
 * Copyright (c) 2019, Joyent, Inc.
 */

/*
 * Collection of common utilities for CTF testing.
 */

#include <strings.h>
#include <libctf.h>
#include "check-common.h"

typedef struct ctftests_lookup_cb {
        ctf_file_t *clc_fp;
        ctf_id_t clc_id;
        const char *clc_name;
} ctftests_lookup_cb_t;
```

*unchanged_portion_omitted_*

```
boolean_t
ctftest_check_descent(const char *symbol, ctf_file_t *fp,
    const check_descent_t *tests, boolean_t quiet)
    const check_descent_t *tests)
{
        ctf_id_t base;
        uint_t layer = 0;

        /*
         * First, find the initial type of the symbol.
         */
        base = ctftest_lookup_symbol(fp, symbol);
        if (base == CTF_ERR) {
                warnx("failed to lookup type for symbol %s", symbol);
                return (B_FALSE);
        }

        while (tests->cd_tname != NULL) {
                ctf_id_t tid;
                int kind;
                ctf_arinfo_t ari;
```



```
                if (base == CTF_ERR) {
                        if (!quiet) {
                                warnx("encountered non-reference type at layer "
                                    "%u while still expecting type %s for "
                                    "symbol %s", layer,
                                    tests->cd_tname, symbol);
                        }
                        warnx("encountered non-reference type at layer %u "
                            "while still expecting type %s for symbol %s",
                            layer, tests->cd_tname, symbol);
                        return (B_FALSE);
                }

                tid = ctftest_lookup_type(fp, tests->cd_tname);
                if (tid == CTF_ERR) {
                        if (!quiet) {
                                warnx("failed to lookup type %s",
                                    tests->cd_tname);
                        }
                        warnx("failed to lookup type %s", tests->cd_tname);
                        return (B_FALSE);
                }

                if (tid != base) {
                        if (!quiet) {
                                warnx("type mismatch at layer %u: found id %u, "
                                    "but expecting type id %u for type %s, "
                                    "symbol %s", layer, base, tid,
                                    tests->cd_tname, symbol);
                        }
                        warnx("type mismatch at layer %u: found id %u, but "
                            "expecting type id %u for type %s, symbol %s",
                            layer, base, tid, tests->cd_tname, symbol);
                        return (B_FALSE);
                }

                kind = ctf_type_kind(fp, base);
                if (kind != tests->cd_kind) {
                        if (!quiet) {
                                warnx("type kind mismatch at layer %u: found "
                                    "kind %u, but expected kind %u for %s, "
                                    "symbol %s", layer, kind, tests->cd_kind,
                                    tests->cd_tname, symbol);
                        }
                        warnx("type kind mismatch at layer %u: found kind %u, "
                            "but expected kind %u for %s, symbol %s", layer,
                            kind, tests->cd_kind, tests->cd_tname, symbol);
                        return (B_FALSE);
                }

                switch (kind) {
                case CTF_K_ARRAY:
                        if (ctf_array_info(fp, base, &ari) == CTF_ERR) {
                                if (!quiet) {
                                        warnx("failed to lookup array info at "
                                            "layer %u for type %s, symbol "
                                            "%s: %s", base, tests->cd_tname,
                                            symbol, ctf_errmsg(ctf_errno(fp)));
                                }
                                warnx("failed to lookup array info at layer "
                                    "%u for type %s, symbol %s: %s", base,
                                    tests->cd_tname, symbol,
                                    ctf_errmsg(ctf_errno(fp)));
                                return (B_FALSE);
                        }
```

```
                        if (tests->cd_nents != ari.ctr_nelems) {
                                if (!quiet) {
                                        warnx("array element mismatch at layer "
                                            "%u for type %s, symbol %s: found "
                                            "%u, expected %u", layer,
                                            tests->cd_tname, symbol,
                                            ari.ctr_nelems, tests->cd_nents);
                                }
                                warnx("array element mismatch at layer %u "
                                    "for type %s, symbol %s: found %u, "
                                    "expected %u", layer, tests->cd_tname,
                                    symbol, ari.ctr_nelems, tests->cd_nents);
                                return (B_FALSE);
                        }

                        tid = ctftest_lookup_type(fp, tests->cd_contents);
                        if (tid == CTF_ERR) {
                                if (!quiet) {
                                        warnx("failed to look up type %s",
                                            tests->cd_contents);
                                }
                                return (B_FALSE);
                        }

                        if (ari.ctr_contents != tid) {
                                if (!quiet) {
                                        warnx("array contents mismatch at "
                                            "layer %u for type %s, symbol %s: "
                                            "found %u, expected %s/%u", layer,
                                            tests->cd_tname, symbol,
                                            ari.ctr_contents,
                                warnx("array contents mismatch at layer %u "
                                    "for type %s, symbol %s: found %u, "
                                    "expected %s/%u", layer, tests->cd_tname,
                                    symbol, ari.ctr_contents,
                                            tests->cd_contents, tid);
                                }

                                return (B_FALSE);
                        }
                        base = ari.ctr_contents;
                        break;
                default:
                        base = ctf_type_reference(fp, base);
                        break;
                }

                tests++;
                layer++;
        }

        if (base != CTF_ERR) {
                if (!quiet) {
                        warnx("found additional type %u in chain, "
                            "but expected no more", base);
                }
                warnx("found additional type %u in chain, but expected no more",
                    base);
                return (B_FALSE);
        }

        return (B_TRUE);
 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    3448 Fri Apr 26 04:01:30 2019
new/usr/src/test/util-tests/tests/ctf/check-common.h
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright 2019, Joyent, Inc.
  13  * Copyright (c) 2019, Joyent, Inc.
  14  */

  16 #ifndef _CHECK_COMMON_H
  17 #define _CHECK_COMMON_H

  19 /*
  20  * Common definitions for the CTF tests
  21  */

  23 #include <stdlib.h>
  24 #include <unistd.h>
  25 #include <libctf.h>
  26 #include <err.h>
  27 #include <strings.h>
  28 #include <sys/param.h>

  30 #ifdef __cplusplus
  31 extern "C" {
  32 #endif

  34 typedef struct check_number {
  35         const char *cn_tname;
  36         uint_t cn_kind;
  37         uint_t cn_flags;
  38         uint_t cn_offset;
  39         uint_t cn_size;
  40 } check_number_t;
______unchanged_portion_omitted_

  90 /*
  91  * Looks up each type and verifies that it matches the expected type.
  92  */
  93 extern boolean_t ctftest_check_numbers(ctf_file_t *, const check_number_t *);

  95 /*
  96  * Looks at each symbol specified and verifies that it matches the expected
  97  * type.
  98  */
  99 extern boolean_t ctftest_check_symbols(ctf_file_t *, const check_symbol_t *);
```



```
 101 /*
 102  * Given a symbol name which refers to a type, walks all the references of that
 103  * type and checks against it with each subsequent entry.
 104  */
 105 extern boolean_t ctftest_check_descent(const char *, ctf_file_t *,
 106     const check_descent_t *, boolean_t);
 106     const check_descent_t *);

 108 /*
 109  * Checks that all of the listed members of an enum are present and have the
 110  * right values.
 111  */
 112 extern boolean_t ctftest_check_enum(const char *, ctf_file_t *,
 113     const check_enum_t *);

 115 /*
 116  * Checks that all of the members of a structure or union are present and have
 117  * the right types and byte offsets. This can be used for either structures or
 118  * unions.
 119  */
 120 extern boolean_t ctftest_check_members(const char *, ctf_file_t *, int, size_t,
 121     const check_member_t *);

 123 /*
 124  * Check that the named function or function pointer has the correct return
 125  * type, arguments, and function flags.
 126  */
 127 extern boolean_t ctftest_check_function(const char *, ctf_file_t *,
 128     const char *, uint_t, uint_t, const char **);
 129 extern boolean_t ctftest_check_fptr(const char *, ctf_file_t *,
 130     const char *, uint_t, uint_t, const char **);

 132 /*
 133  * Determine whether or not we have a duplicate type or not based on its name.
 134  */
 135 extern boolean_t ctftest_duplicates(ctf_file_t *);

 137 #ifdef __cplusplus
 138 }
______unchanged_portion_omitted_
```

```
**********************************************************
    2874 Fri Apr 26 04:01:30 2019
new/usr/src/test/util-tests/tests/ctf/check-enum.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2019, Joyent, Inc.
 * Copyright (c) 2019, Joyent, Inc.
 */

/*
 * Check that we properly handle enums.
 */

#include "check-common.h"

static check_symbol_t check_syms[] = {
        { "ff6", "enum ff6" },
        { "ff10", "ff10_t" },
        { NULL }
};
```

*unchanged_portion_omitted_*

```
int
main(int argc, char *argv[])
{
        int i, ret = 0;

        if (argc < 2) {
                errx(EXIT_FAILURE, "missing test files");
        }

        for (i = 1; i < argc; i++) {
                ctf_file_t *fp;
                uint_t d;

                if ((fp = ctf_open(argv[i], &ret)) == NULL) {
                        warnx("failed to open %s: %s", argv[i],
                            ctf_errmsg(ret));
                        ret = EXIT_FAILURE;
                        continue;
                }
                if (!ctftest_check_symbols(fp, check_syms))
                        ret = EXIT_FAILURE;
                for (d = 0; descents[d].cdt_sym != NULL; d++) {
                        if (!ctftest_check_descent(descents[d].cdt_sym, fp,
                            descents[d].cdt_tests, B_FALSE)) {
                            descents[d].cdt_tests)) {
```



```
                                ret = EXIT_FAILURE;
                        }
                }

                for (d = 0; enums[d].cet_type != NULL; d++) {
                        if (!ctftest_check_enum(enums[d].cet_type, fp,
                            enums[d].cet_tests)) {
                                ret = EXIT_FAILURE;
                        }
                }
                ctf_close(fp);
        }

        return (ret);

}
```

*unchanged_portion_omitted_*

```
**********************************************************
    3201 Fri Apr 26 04:01:30 2019
new/usr/src/test/util-tests/tests/ctf/check-forward.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2019, Joyent, Inc.
 * Copyright (c) 2019, Joyent, Inc.
 */

/*
 * Verify that we can properly handle forward declarations.
 *
 * In test-forward.c barp is declared as a union, not a struct. However, today
 * the CTF tooling does not contain enough information to know whether a forward
 * declaration was for a struct or a union, only that it was a forward.
 * Therefore, the type printing information assumes at the moment that the type
 * is a struct. In a future revision of the CTF type data, we should encode this
 * information in the equivalent of ctt_info so we can properly distinguish
 * between these.
 */

#include "check-common.h"

static check_symbol_t check_syms[] = {
        { "forward", "struct forward" },
        { "foop", "struct foo *" },
        { "barp", "struct bar *" },
        { "bazp", "enum baz *" },
        { NULL }
};
```

*unchanged_portion_omitted_*

```
int
main(int argc, char *argv[])
{
        int i, ret = 0;

        if (argc < 2) {
                errx(EXIT_FAILURE, "missing test files");
        }

        for (i = 1; i < argc; i++) {
                ctf_file_t *fp;
                uint_t j;

                if ((fp = ctf_open(argv[i], &ret)) == NULL) {
                        warnx("failed to open %s: %s", argv[i],
                            ctf_errmsg(ret));
```



```
                        ret = EXIT_FAILURE;
                        continue;
                }

                if (!ctftest_check_symbols(fp, check_syms))
                        ret = EXIT_FAILURE;

                for (j = 0; descents[j].cdt_sym != NULL; j++) {
                        if (!ctftest_check_descent(descents[j].cdt_sym, fp,
                            descents[j].cdt_tests, B_FALSE)) {
                            descents[j].cdt_tests)) {
                                ret = EXIT_FAILURE;
                        }
                }


                for (j = 0; members[j].cmt_type != NULL; j++) {
                        if (!ctftest_check_members(members[j].cmt_type, fp,
                            members[j].cmt_kind, members[j].cmt_size,
                            members[j].cmt_members)) {
                                ret = EXIT_FAILURE;
                        }
                }

                ctf_close(fp);
        }

        return (ret);
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    9060 Fri Apr 26 04:01:31 2019
new/usr/src/test/util-tests/tests/ctf/check-qualifiers.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2019, Joyent, Inc.
 */

/*
 * Check qualifier encoding. Note that the needed_qualifier() workaround applies
 * to most of these.
 */

#include "check-common.h"

static check_descent_t check_descent_const_union_array_gcc4[] = {
        { "const union const_union [5]", CTF_K_CONST },
        { "union const_union [5]", CTF_K_ARRAY, "union const_union", 5 },
        { "union const_union", CTF_K_UNION },
        { NULL }
};

static check_descent_t check_descent_const_union_array_gcc7[] = {
        { "const union const_union [5]", CTF_K_ARRAY,
            "const union const_union", 5 },
        { "const union const_union", CTF_K_CONST },
        { "union const_union", CTF_K_UNION },
        { NULL }
};

static check_descent_test_t alt_descents_const_union_array[] = {
        { "const_union_array", check_descent_const_union_array_gcc4 },
        { "const_union_array", check_descent_const_union_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_const_struct_array_gcc4[] = {
        { "const struct const_struct [7]", CTF_K_CONST },
        { "struct const_struct [7]", CTF_K_ARRAY, "struct const_struct", 7 },
        { "struct const_struct", CTF_K_STRUCT },
        { NULL }
};

static check_descent_t check_descent_const_struct_array_gcc7[] = {
        { "const struct const_struct [7]", CTF_K_ARRAY,
            "const struct const_struct", 7 },
        { "const struct const_struct", CTF_K_CONST },
```



```
        { "struct const_struct", CTF_K_STRUCT },
        { NULL }
};

static check_descent_test_t alt_descents_const_struct_array[] = {
        { "const_struct_array", check_descent_const_struct_array_gcc4 },
        { "const_struct_array", check_descent_const_struct_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_volatile_struct_array_gcc4[] = {
        { "volatile struct volatile_struct [9]", CTF_K_VOLATILE },
        { "struct volatile_struct [9]", CTF_K_ARRAY,
            "struct volatile_struct", 9 },
        { "struct volatile_struct", CTF_K_STRUCT },
        { NULL }
};

static check_descent_t check_descent_volatile_struct_array_gcc7[] = {
        { "volatile struct volatile_struct [9]", CTF_K_ARRAY,
            "volatile struct volatile_struct", 9 },
        { "volatile struct volatile_struct", CTF_K_VOLATILE },
        { "struct volatile_struct", CTF_K_STRUCT },
        { NULL }
};

static check_descent_test_t alt_descents_volatile_struct_array[] = {
        { "volatile_struct_array", check_descent_volatile_struct_array_gcc4 },
        { "volatile_struct_array", check_descent_volatile_struct_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_c_int_array_gcc4[] = {
        { "const int [11]", CTF_K_CONST },
        { "int [11]", CTF_K_ARRAY, "int", 11 },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_c_int_array_gcc7[] = {
        { "const int [11]", CTF_K_ARRAY, "const int", 11 },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_test_t alt_descents_c_int_array[] = {
        { "c_int_array", check_descent_c_int_array_gcc4 },
        { "c_int_array", check_descent_c_int_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_cv_int_array_gcc4[] = {
        { "const volatile int [13]", CTF_K_CONST },
        { "volatile int [13]", CTF_K_VOLATILE },
        { "int [13]", CTF_K_ARRAY, "int", 13 },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_cv_int_array_gcc7[] = {
        { "volatile const int [13]", CTF_K_ARRAY, "volatile const int", 13 },
        { "volatile const int", CTF_K_VOLATILE },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
```

```
};

static check_descent_test_t alt_descents_cv_int_array[] = {
        { "cv_int_array", check_descent_cv_int_array_gcc4 },
        { "cv_int_array", check_descent_cv_int_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_vc_int_array_gcc4[] = {
        { "const volatile int [15]", CTF_K_CONST },
        { "volatile int [15]", CTF_K_VOLATILE },
        { "int [15]", CTF_K_ARRAY, "int", 15 },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_vc_int_array_gcc7[] = {
        { "volatile const int [15]", CTF_K_ARRAY, "volatile const int", 15 },
        { "volatile const int", CTF_K_VOLATILE },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_test_t alt_descents_vc_int_array[] = {
        { "vc_int_array", check_descent_vc_int_array_gcc4 },
        { "vc_int_array", check_descent_vc_int_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_vc_int_array2_gcc4[] = {
        { "const volatile int [17]", CTF_K_CONST },
        { "volatile int [17]", CTF_K_VOLATILE },
        { "int [17]", CTF_K_ARRAY, "int", 17 },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_vc_int_array2_gcc7[] = {
        { "volatile const int [17]", CTF_K_ARRAY, "volatile const int", 17 },
        { "volatile const int", CTF_K_VOLATILE },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_test_t alt_descents_vc_int_array2[] = {
        { "vc_int_array2", check_descent_vc_int_array2_gcc4 },
        { "vc_int_array2", check_descent_vc_int_array2_gcc7 },
        { NULL }
};

static check_descent_t check_descent_c_2d_array_gcc4[] = {
        { "const int [4][2]", CTF_K_CONST },
        { "int [4][2]", CTF_K_ARRAY, "int [2]", 4 },
        { "int [2]", CTF_K_ARRAY, "int", 2 },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_c_2d_array_gcc7[] = {
        { "const int [4][2]", CTF_K_ARRAY, "const int [2]", 4 },
        { "const int [2]", CTF_K_ARRAY, "const int", 2 },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
```



```
};

static check_descent_test_t alt_descents_c_2d_array[] = {
        { "c_2d_array", check_descent_c_2d_array_gcc4 },
        { "c_2d_array", check_descent_c_2d_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_cv_3d_array_gcc4[] = {
        { "const volatile int [3][2][1]", CTF_K_CONST },
        { "volatile int [3][2][1]", CTF_K_VOLATILE },
        { "int [3][2][1]", CTF_K_ARRAY, "int [2][1]", 3 },
        { "int [2][1]", CTF_K_ARRAY, "int [1]", 2 },
        { "int [1]", CTF_K_ARRAY, "int", 1 },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_cv_3d_array_gcc7[] = {
        { "volatile const int [3][2][1]", CTF_K_ARRAY,
          "volatile const int [2][1]", 3 },
        { "volatile const int [2][1]", CTF_K_ARRAY,
          "volatile const int [1]", 2 },
        { "volatile const int [1]", CTF_K_ARRAY, "volatile const int", 1 },
        { "volatile const int", CTF_K_VOLATILE },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_test_t alt_descents_cv_3d_array[] = {
        { "cv_3d_array", check_descent_cv_3d_array_gcc4 },
        { "cv_3d_array", check_descent_cv_3d_array_gcc7 },
        { NULL }
};

static check_descent_t check_descent_ptr_to_const_int[] = {
        { "const int *", CTF_K_POINTER },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_test_t alt_descents_ptr_to_const_int[] = {
        { "ptr_to_const_int", check_descent_ptr_to_const_int },
        { NULL }
};

static check_descent_t check_descent_const_ptr_to_int[] = {
        { "int *const", CTF_K_CONST },
        { "int *", CTF_K_POINTER },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_test_t alt_descents_const_ptr_to_int[] = {
        { "const_ptr_to_int", check_descent_const_ptr_to_int },
        { NULL }
};

static check_descent_t check_descent_const_ptr_to_const_int[] = {
        { "const int *const", CTF_K_CONST },
        { "const int *", CTF_K_POINTER },
        { "const int", CTF_K_CONST },
        { "int", CTF_K_INTEGER },
        { NULL }
```

```
253 };

255 static check_descent_test_t alt_descents_const_ptr_to_const_int[] = {
256         { "const_ptr_to_const_int", check_descent_const_ptr_to_const_int },
257         { NULL }
258 };

260 static check_descent_test_t *alt_descents[] = {
261         alt_descents_const_union_array,
262         alt_descents_const_struct_array,
263         alt_descents_volatile_struct_array,
264         alt_descents_c_int_array,
265         alt_descents_cv_int_array,
266         alt_descents_vc_int_array,
267         alt_descents_vc_int_array2,
268         alt_descents_c_2d_array,
269         alt_descents_cv_3d_array,
270         alt_descents_ptr_to_const_int,
271         alt_descents_const_ptr_to_int,
272         alt_descents_const_ptr_to_const_int,
273         NULL
274 };

276 int
277 main(int argc, char *argv[])
278 {
279         int i, ret = 0;

281         if (argc < 2) {
282                 errx(EXIT_FAILURE, "missing test files");
283         }

285         for (i = 1; i < argc; i++) {
286                 ctf_file_t *fp;

288                 if ((fp = ctf_open(argv[i], &ret)) == NULL) {
289                         warnx("failed to open %s: %s", argv[i],
290                             ctf_errmsg(ret));
291                         ret = EXIT_FAILURE;
292                         continue;
293                 }

295                 for (uint_t j = 0; alt_descents[j] != NULL; j++) {
296                         check_descent_test_t *descents = alt_descents[j];
297                         int alt_ok = 0;

299                         for (uint_t k = 0; descents[k].cdt_sym != NULL; k++) {
300                                 if (ctftest_check_descent(descents[k].cdt_sym,
301                                     fp, descents[k].cdt_tests, B_TRUE)) {
302                                         alt_ok = 1;
303                                         break;
304                                 }
305                         }

307                         if (!alt_ok) {
308                                 warnx("all descents failed for %s",
309                                     descents[0].cdt_sym);
310                                 ret = EXIT_FAILURE;
311                         }
312                 }

314                 ctf_close(fp);
315         }

317         return (ret);
318 }
```

```
**********************************************************
    5682 Fri Apr 26 04:01:31 2019
new/usr/src/test/util-tests/tests/ctf/check-reference.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
  1 /*
  2  * This file and its contents are supplied under the terms of the
  3  * Common Development and Distribution License ("CDDL"), version 1.0.
  4  * You may only use this file in accordance with the terms of version
  5  * 1.0 of the CDDL.
  6  *
  7  * A full copy of the text of the CDDL should have accompanied this
  8  * source.  A copy of the CDDL is also available via the Internet at
  9  * http://www.illumos.org/license/CDDL.
 10  */

 12 /*
 13  * Copyright 2019, Joyent, Inc.
 13  * Copyright (c) 2019, Joyent, Inc.
 14  */

 16 /*
 17  * Check that we properly understand reference types and can walk through them
 18  * as well as generate them.
 19  */

 21 #include "check-common.h"

 23 static check_number_t check_base[] = {
 24         { "char", CTF_K_INTEGER, CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 },
 25         { "int", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 32 },
 26         { "float", CTF_K_FLOAT, CTF_FP_SINGLE, 0, 32 },
 27         { NULL }
 28 };
_____unchanged_portion_omitted_

 134 static check_descent_t check_descent_cvh[] = {
 135         { "const volatile foo_t *", CTF_K_POINTER },
 136         { "const volatile foo_t", CTF_K_CONST },
 137         { "volatile foo_t", CTF_K_VOLATILE },
 138         { "foo_t", CTF_K_TYPEDEF },
 139         { "int *const *", CTF_K_POINTER },
 140         { "int *const", CTF_K_CONST },
 141         { "int *", CTF_K_POINTER },
 142         { "int", CTF_K_INTEGER },
 143         { NULL }
 144 };

 134 static check_descent_test_t descents[] = {
 135         { "aa", check_descent_aa },
 136         { "b", check_descent_b },
 137         { "c", check_descent_c },
 138         { "d", check_descent_d },
 139         { "dd", check_descent_dd },
 140         { "ddd", check_descent_ddd },
 141         { "e", check_descent_e },
 142         { "ce", check_descent_ce },
 143         { "ve", check_descent_ve },
 144         { "cve", check_descent_cve },
```



```
 145         { "f", check_descent_f },
 146         { "g", check_descent_g },
 159         { "cvh", check_descent_cvh },
 147         { NULL }
 148 };

 150 static check_descent_t check_descent_cvh_gcc4[] = {
 151         { "const volatile foo_t *", CTF_K_POINTER },
 152         { "const volatile foo_t", CTF_K_CONST },
 153         { "volatile foo_t", CTF_K_VOLATILE },
 154         { "foo_t", CTF_K_TYPEDEF },
 155         { "int *const *", CTF_K_POINTER },
 156         { "int *const", CTF_K_CONST },
 157         { "int *", CTF_K_POINTER },
 158         { "int", CTF_K_INTEGER },
 159         { NULL }
 160 };

 162 static check_descent_t check_descent_cvh_gcc7[] = {
 163         { "volatile const foo_t *", CTF_K_POINTER },
 164         { "volatile const foo_t", CTF_K_VOLATILE },
 165         { "const foo_t", CTF_K_CONST },
 166         { "foo_t", CTF_K_TYPEDEF },
 167         { "int *const *", CTF_K_POINTER },
 168         { "int *const", CTF_K_CONST },
 169         { "int *", CTF_K_POINTER },
 170         { "int", CTF_K_INTEGER },
 171         { NULL }
 172 };

 174 /*
 175  * GCC versions differ in how they order qualifiers, which is a shame for
 176  * round-tripping; but as they're clearly both valid, we should cope.  We'll
 177  * just insist that at least one of these checks passes.
 178  */
 179 static check_descent_test_t alt_descents[] = {
 180         { "cvh", check_descent_cvh_gcc4 },
 181         { "cvh", check_descent_cvh_gcc7 },
 182 };

 184 int
 185 main(int argc, char *argv[])
 186 {
 187         int i, ret = 0;

 189         if (argc < 2) {
 190                 errx(EXIT_FAILURE, "missing test files");
 191         }

 193         for (i = 1; i < argc; i++) {
 194                 ctf_file_t *fp;
 195                 int alt_ok = 0;
 196                 uint_t d;

 198                 if ((fp = ctf_open(argv[i], &ret)) == NULL) {
 199                         warnx("failed to open %s: %s", argv[i],
 200                             ctf_errmsg(ret));
 201                         ret = EXIT_FAILURE;
 202                         continue;
 203                 }

 205                 if (!ctftest_check_numbers(fp, check_base))
 206                         ret = EXIT_FAILURE;
 207                 if (!ctftest_check_symbols(fp, check_syms))
 208                         ret = EXIT_FAILURE;
 209                 for (d = 0; descents[d].cdt_sym != NULL; d++) {
```

```
                        if (!ctftest_check_descent(descents[d].cdt_sym, fp,
                            descents[d].cdt_tests, B_FALSE)) {
                            descents[d].cdt_tests)) {
                                ret = EXIT_FAILURE;
                        }
                }

                for (d = 0; alt_descents[d].cdt_sym != NULL; d++) {
                        if (ctftest_check_descent(alt_descents[d].cdt_sym, fp,
                            alt_descents[d].cdt_tests, B_TRUE)) {
                                alt_ok = 1;
                                break;
                        }
                }

                if (!alt_ok) {
                        warnx("all descents failed for %s",
                            alt_descents[0].cdt_sym);
                        ret = EXIT_FAILURE;
                }

                ctf_close(fp);
        }

        return (ret);
}
```

*unchanged_portion_omitted_*

```
**********************************************************
   12237 Fri Apr 26 04:01:31 2019
new/usr/src/test/util-tests/tests/ctf/check-sou.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2019, Joyent, Inc.
 * Copyright (c) 2019, Joyent, Inc.
 */

/*
 * Check that we properly handle structures and unions.
 */

#include "check-common.h"

static check_number_t check_bitfields[] = {
#ifdef  TARGET_LP64
        { "unsigned long:1", CTF_K_INTEGER, 0, 0, 1 },
        { "unsigned long:2", CTF_K_INTEGER,  0, 0, 2 },
        { "unsigned long:4", CTF_K_INTEGER,  0, 0, 4 },
        { "unsigned long:5", CTF_K_INTEGER,  0, 0, 5 },
        { "unsigned long:8", CTF_K_INTEGER,  0, 0, 8 },
        { "unsigned long:16", CTF_K_INTEGER,  0, 0, 16 },
        { "unsigned long:19", CTF_K_INTEGER,  0, 0, 19 },
        { "unsigned long:32", CTF_K_INTEGER,  0, 0, 32 },
#else
        { "unsigned long long:1", CTF_K_INTEGER, 0, 0, 1 },
        { "unsigned long long:2", CTF_K_INTEGER,  0, 0, 2 },
        { "unsigned long long:4", CTF_K_INTEGER,  0, 0, 4 },
        { "unsigned long long:5", CTF_K_INTEGER,  0, 0, 5 },
        { "unsigned long long:8", CTF_K_INTEGER,  0, 0, 8 },
        { "unsigned long long:16", CTF_K_INTEGER,  0, 0, 16 },
        { "unsigned long long:19", CTF_K_INTEGER,  0, 0, 19 },
        { "unsigned long long:32", CTF_K_INTEGER,  0, 0, 32 },
#endif
        { "unsigned short:1", CTF_K_INTEGER, 0, 0, 1 },
        { "unsigned int:7", CTF_K_INTEGER, 0, 0, 7 },
        { "unsigned int:32", CTF_K_INTEGER, 0, 0, 32 },
        { "int:3", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 3 },
        { NULL }
};
_____unchanged_portion_omitted_

static check_descent_test_t descents[] = {
        { "head", check_descent_head },
        { "forward", check_descent_forward },
        { NULL }
```



```
};

static check_descent_t check_descent_regress_gcc4[] = {
static check_descent_t check_descent_regress[] = {
        { "const union regress [9]", CTF_K_CONST },
        { "union regress [9]", CTF_K_ARRAY, "union regress", 9 },
        { "union regress", CTF_K_UNION },
        { NULL }
};

static check_descent_t check_descent_regress_gcc7[] = {
        { "const union regress [9]", CTF_K_ARRAY, "const union regress", 9 },
        { "const union regress", CTF_K_CONST },
        { "union regress", CTF_K_UNION },
static check_descent_test_t descents[] = {
        { "head", check_descent_head },
        { "forward", check_descent_forward },
        { "regress", check_descent_regress },
        { NULL }
};

/*
 * See needed_array_qualifier(): applying this fix means the qualifier order is
 * different between GCC versions. Accept either form.
 */
static check_descent_test_t alt_descents[] = {
        { "regress", check_descent_regress_gcc4 },
        { "regress", check_descent_regress_gcc7 },
        { NULL }
};

int
main(int argc, char *argv[])
{
        int i, ret = 0;

        if (argc < 2) {
                errx(EXIT_FAILURE, "missing test files");
        }

        for (i = 1; i < argc; i++) {
                ctf_file_t *fp;
                int alt_ok = 0;
                uint_t j;

                if ((fp = ctf_open(argv[i], &ret)) == NULL) {
                        warnx("failed to open %s: %s", argv[i],
                            ctf_errmsg(ret));
                        ret = EXIT_FAILURE;
                        continue;
                }

                if (!ctftest_check_numbers(fp, check_bitfields))
                        ret = EXIT_FAILURE;
                if (!ctftest_check_symbols(fp, check_syms))
                        ret = EXIT_FAILURE;
                for (j = 0; descents[j].cdt_sym != NULL; j++) {
                        if (!ctftest_check_descent(descents[j].cdt_sym, fp,
                            descents[j].cdt_tests, B_FALSE)) {
                            descents[j].cdt_tests)) {
                                ret = EXIT_FAILURE;
                        }
                }

                for (j = 0; alt_descents[j].cdt_sym != NULL; j++) {
                        if (ctftest_check_descent(alt_descents[j].cdt_sym, fp,
```

```
412                             alt_descents[j].cdt_tests, B_TRUE)) {
413                                 alt_ok = 1;
414                                 break;
415                         }
416                 }

418                 if (!alt_ok) {
419                         warnx("all descents failed for %s",
420                             alt_descents[0].cdt_sym);
421                         ret = EXIT_FAILURE;
422                 }

424                 for (j = 0; members[j].cmt_type != NULL; j++) {
425                         if (!ctftest_check_members(members[j].cmt_type, fp,
426                             members[j].cmt_kind, members[j].cmt_size,
427                             members[j].cmt_members)) {
428                                 ret = EXIT_FAILURE;
429                         }
430                 }

432                 ctf_close(fp);
433         }

435         return (ret);
436 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
     663 Fri Apr 26 04:01:31 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-weak/test-merge-weak.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
   1 /*
   2  * This file and its contents are supplied under the terms of the
   3  * Common Development and Distribution License ("CDDL"), version 1.0.
   4  * You may only use this file in accordance with the terms of version
   5  * 1.0 of the CDDL.
   6  *
   7  * A full copy of the text of the CDDL should have accompanied this
   8  * source.  A copy of the CDDL is also available via the Internet at
   9  * http://www.illumos.org/license/CDDL.
  10  */

  12 /*
  13  * Copyright 2019, Joyent, Inc.
  13  * Copyright (c) 2019, Joyent, Inc.
  14  */

  16 #include <stdlib.h>

  18 #pragma weak mumble = _mumble
  19 #pragma weak foo = _foo

  21 int _foo = 5;

  23 int
  24 _mumble(void)
  25 {
  26         return ((int)arc4random());
  27 }

  29 extern int mumble(void);

  31 int
  32 main(void)
  33 {
  34         return (mumble());
  35 };
```

_____unchanged_portion_omitted_

```
**********************************************************
    1020 Fri Apr 26 04:01:31 2019
new/usr/src/test/util-tests/tests/ctf/test-qualifiers.c
10823 should ignore DW_TAG_subprogram with DW_AT_declaration tags
10824 GCC7-derived CTF can double qualifiers on arrays
10825 ctfdump -c drops last type
10826 ctfdump -c goes off the rails with a missing parent
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Jason King <jason.king@joyent.com>
Approved by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2019, Joyent, Inc.
 */

/*
 * Make sure that we're encoding qualifiers correctly.
 */

const union const_union {
        int i;
} const_union_array[5];

const struct const_struct {
        int i;
} const_struct_array[7];

volatile struct volatile_struct {
        int i;
} volatile_struct_array[9];

const int c_int_array[11];
const volatile int cv_int_array[13];
volatile const int vc_int_array[15];
volatile int const vc_int_array2[17];

const int c_2d_array[4][2];
const volatile int cv_3d_array[3][2][1];

const int *ptr_to_const_int;
int * const const_ptr_to_int;
const int * const const_ptr_to_const_int;
```