```
**********************************************************
   33409 Tue Apr 23 05:49:14 2019
new/usr/src/cmd/ctfdump/ctfdump.c
10822 ctfdump -c should include non-root types
Reviewed by: Robert Mustacchi <rm@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 (c) 2019, Joyent, Inc.
  13  * Copyright (c) 2018, 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_

 971 static void
 972 ctfdump_source(void)
 973 {
 974         ulong_t nr_syms = ctf_nr_syms(g_fp);
 975         ctf_id_t max_id = ctf_max_id(g_fp);
 976         size_t count = 0;
```



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

 980         if ((idnames = calloc(max_id + 1, sizeof (idnames[0]))) == NULL) {
 981                 ctfdump_fatal("failed to alloc idnames: %s\n",
 982                     strerror(errno));
 983         }

 985         if (ctf_type_iter(g_fp, B_TRUE, ctfsrc_collect_types_cb,
 985         if (ctf_type_iter(g_fp, B_FALSE, ctfsrc_collect_types_cb,
 986             idnames) == CTF_ERR) {
 987                 warnx("failed to collect types: %s",
 988                     ctf_errmsg(ctf_errno(g_fp)));
 989                 g_exit = 1;
 990         }

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

 994         for (size_t i = 0; i < max_id; i++) {
 995                 if (idnames[i].ci_id != 0)
 996                         ctfsrc_type(idnames[i].ci_id, idnames[i].ci_name);
 997         }

 999         free(idnames);

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

1003         if ((idnames = calloc(nr_syms, sizeof (idnames[0]))) == NULL) {
1004                 ctfdump_fatal("failed to alloc idnames: %s\n",
1005                     strerror(errno));
1006         }

1008         if (ctf_object_iter(g_fp, ctfsrc_collect_objects_cb,
1009             &count) == CTF_ERR) {
1010                 warnx("failed to collect objects: %s",
1011                     ctf_errmsg(ctf_errno(g_fp)));
1012                 g_exit = 1;
1013         }

1015         qsort(idnames, count, sizeof (ctf_idname_t), idname_compare);

1017         for (size_t i = 0; i < count; i++)
1018                 ctfsrc_object(idnames[i].ci_id, idnames[i].ci_name);

1020         free(idnames);

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

1024         if ((idnames = calloc(nr_syms, sizeof (idnames[0]))) == NULL) {
1025                 ctfdump_fatal("failed to alloc idnames: %s\n",
1026                     strerror(errno));
1027         }

1029         count = 0;

1031         if (ctf_function_iter(g_fp, ctfsrc_collect_functions_cb,
1032             &count) == CTF_ERR) {
1033                 warnx("failed to collect functions: %s",
1034                     ctf_errmsg(ctf_errno(g_fp)));
1035                 g_exit = 1;
1036         }

1038         qsort(idnames, count, sizeof (ctf_idname_t), idname_compare);

1040         for (size_t i = 0; i < count; i++)
1041                 ctfsrc_function(&idnames[i]);
```

```
1043         free(idnames);
1044 }
```

_____***unchanged_portion_omitted_***