****
787 Tue Apr 23 05:35:48 2019
new/usr/src/test/util-tests/tests/Makefile
10814 Want primordial CTF test suite
Reviewed 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 Garrett D'Amore <garrett@damore.org>
# Copyright 2014 Nexenta Systems, Inc. All rights reserved.
# Copyright 2017 Jason King
# Copyright 2019 Joyent, Inc.
#

SUBDIRS = date dis dladm iconv libnvpair_json libsff printf xargs grep_xpg4
SUBDIRS += demangle mergeq workq chown ctf
SUBDIRS += demangle mergeq workq chown

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

```
**********************************************************
    3279 Tue Apr 23 05:35:48 2019
new/usr/src/test/util-tests/tests/ctf/Makefile
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
#

include $(SRC)/Makefile.master

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

SCRIPTS =       ctftest.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-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-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

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)
$(ROOTCHECKS): $(TESTDIR) $(CHECKS)
$(ROOTSCRIPTS): $(TESTDIR) $(SCRIPTS)

$(TESTDIR):
        $(INS.dir)

$(ROOTMAKEDIRS):
        $(INS.dir)

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

$(TESTDIR)/%: %.ksh
        $(INS.rename)
```

```
127 %.o: %.c
128         $(COMPILE.c) -o $@ $<
129         $(POST_PROCESS_O)

131 %.32.o: %.c
132         $(COMPILE.c) -o $@ $<
133         $(POST_PROCESS_O)

135 %.64.o: %.c
136         $(COMPILE.c) -DTARGET_LP64 -o $@ $<
137         $(POST_PROCESS_O)

139 %-32: %.32.o
140         $(LINK.c) -o $@ $< $(COMMON_OBJS) $(LDLIBS)
141         $(POST_PROCESS)

143 %-64: %.64.o
144         $(LINK.c) -o $@ $< $(COMMON_OBJS) $(LDLIBS)
145         $(POST_PROCESS)

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

```
**********************************************************
    2407 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/Makefile.ctftest.com
10814 Want primordial CTF test suite
Reviewed 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.
#

#
# This Makefile is installed onto the target system and is used as part
# of the running tests. It is not used as part of the build.
#
# This makefile could be simplified substantially. However, it does
# everything explicitly to try and work with a wide variety of different
# makes.
#
# The following values should be passed in by the invoker of the
# Makefile:
#
#       CC              C Compiler to use
#       CFLAGS32        32-bit CFLAGS
#       CFLAGS64        64-bit CFLAGS
#       CTFCONVERT      Path to ctfconvert
#       CTFMERGE        Path to ctfmerge
#       DEBUGFLAGS      The set of debug flags to use
#       BUILDDIR        Directory things should be built in
#       CHECK32         Program to check 32-bit output
#       CHECK64         Program to check 64-bit output
#
# The following values should be set before building this:
#
#       TEST            The name of the test program
#       OBJS_C_32       32-bit convert objects
#       OBJS_C_64       64-bit convert objects
#       OBJS_M_32       32-bit merge objects
#       OBJS_M_64       64-bit merge objects
#

CONV32 =        $(BUILDDIR)/$(TEST)-32c
CONV64 =        $(BUILDDIR)/$(TEST)-64c
MERGE32 =       $(BUILDDIR)/$(TEST)-32m
MERGE64 =       $(BUILDDIR)/$(TEST)-64m

BINS =          $(CONV32) \
                $(CONV64) \
                $(MERGE32) \
                $(MERGE64)

build: $(BINS)

$(BUILDDIR)/%.32.c.o: %.c
        $(CC) $(CFLAGS32) $(DEBUGFLAGS) -o $@ -c $<
```



```
$(BUILDDIR)/%.64.c.o: %.c
        $(CC) $(CFLAGS64) $(DEBUGFLAGS) -o $@ -c $<

$(BUILDDIR)/%.32.m.o: %.c
        $(CC) $(CFLAGS32) $(DEBUGFLAGS) -o $@ -c $<
        $(CTFCONVERT) $@

$(BUILDDIR)/%.64.m.o: %.c
        $(CC) $(CFLAGS64) $(DEBUGFLAGS) -o $@ -c $<
        $(CTFCONVERT) $@

$(CONV32): $(OBJS_C_32)
        $(CC) $(CFLAGS32) $(DEBUGFLAGS) -o $@ $(OBJS_C_32)
        $(CTFCONVERT) $@

$(CONV64): $(OBJS_C_64)
        $(CC) $(CFLAGS64) $(DEBUGFLAGS) -o $@ $(OBJS_C_64)
        $(CTFCONVERT) $@

$(MERGE32): $(OBJS_M_32)
        $(CC) $(CFLAGS32) $(DEBUGFLAGS) -o $@ $(OBJS_M_32)
        $(CTFMERGE) -t -o $@ $(OBJS_M_32)

$(MERGE64): $(OBJS_M_64)
        $(CC) $(CFLAGS64) $(DEBUGFLAGS) -o $@ $(OBJS_M_64)
        $(CTFMERGE) -t -o $@ $(OBJS_M_64)

run-test:
        $(CHECK32) $(CONV32)
        $(CHECK64) $(CONV64)
        $(CHECK32) $(MERGE32)
        $(CHECK64) $(MERGE64)
```

```
**********************************************************
    1867 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/README
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
#

CTF Tests
---------

This directory contains a series of tests for the Compact C Type Format
(CTF). For each test program, there is a corresponding C program that
goes through and checks the CTF for various aspects. Due to the fact
that the CTF generated by compilers can change slightly, the tests have
been designed this way to try and make it work with as wide a variety of
programs as possible.

The test suite requires the following:

1. make
2. C Compiler (defaults to gcc)
3. A copy of ctfconvert

The source for a given program will be compiled on the target system and
then converted. This allows us to try the CTF tools against a wide
variety of different compilers or DWARF standards.

Caveats
-------

Right now the tests only pass when using gcc 4.x. The following are
known issues with the tests:

1. gcc7+ generates some different DWARF ordering, which causes some
tests to spuriously fail. These tests should be improved.

2. There are cases where gcc7+ appears to attribute things as being const
twice in DWARF which throw off the tests. The CTF tools likely should
work around this if we confirm that this is intentional.

3. Many tests will cause clang not to emit DWARF information because
clang infers that they cannot be used. The tests should be cleaned up in
these cases.

4. clang generated DWARF can confuse the CTF tools. The tools should be
fixed and additional regression tests should be added.
```

```
**********************************************************
    2897 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-array.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Check that we properly generate basic nested arrays.
 */

#include "check-common.h"

static check_number_t check_base[] = {
        { "char", CTF_K_INTEGER, CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 },
        { "int", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 32 },
        { "double", CTF_K_FLOAT, CTF_FP_DOUBLE, 0, 64 },
        { NULL }
};

static check_symbol_t check_syms[] = {
        { "a", "int [3]" },
        { "b", "double [42]" },
        { "c", "const char *[2]" },
        { "d", "int [4][5]" },
        { "e", "int [4][5][6]" },
        { "f", "int [4][5][6][7]" },
        { "g", "int [4][5][6][7][8]" },
        { "h", "int [4][5][6][7][8][9]" },
        { "i", "int [4][5][6][7][8][9][10]" },
        { NULL }
};

static check_descent_t check_array_a[] = {
        { "int [3]", CTF_K_ARRAY, "int", 3 },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_array_b[] = {
        { "double [42]", CTF_K_ARRAY, "double", 42 },
        { "double", CTF_K_FLOAT },
        { NULL }
};

static check_descent_t check_array_c[] = {
        { "const char *[2]", CTF_K_ARRAY, "const char *", 2 },
        { "const char *", CTF_K_POINTER },
        { "const char", CTF_K_CONST },
        { "char", CTF_K_INTEGER },
        { NULL }
};
```



```
static check_descent_t check_array_i[] = {
        { "int [4][5][6][7][8][9][10]", CTF_K_ARRAY,
            "int [5][6][7][8][9][10]", 4 },
        { "int [5][6][7][8][9][10]", CTF_K_ARRAY, "int [6][7][8][9][10]", 5 },
        { "int [6][7][8][9][10]", CTF_K_ARRAY, "int [7][8][9][10]", 6 },
        { "int [7][8][9][10]", CTF_K_ARRAY, "int [8][9][10]", 7 },
        { "int [8][9][10]", CTF_K_ARRAY, "int [9][10]", 8 },
        { "int [9][10]", CTF_K_ARRAY, "int [10]", 9 },
        { "int [10]", CTF_K_ARRAY, "int", 10 },
        { "int", CTF_K_INTEGER },
        { NULL },
};

static check_descent_test_t descents[] = {
        { "a", check_array_a },
        { "b", check_array_b },
        { "c", check_array_c },
        { "i", check_array_i },
        { 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;
                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_numbers(fp, check_base))
                        ret = EXIT_FAILURE;
                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)) {
                                ret = EXIT_FAILURE;
                        }
                }
                ctf_close(fp);
        }

        return (ret);
}
```

```
**********************************************************
   18568 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-common.c
10814 Want primordial CTF test suite
Reviewed 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) 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;

typedef struct ctftest_member_cb {
        ctf_file_t *cmc_fp;
        const check_member_t *cmc_members;
        const char *cmc_name;
} ctftest_member_cb_t;

static int
ctftest_lookup_type_cb(ctf_id_t id, boolean_t root, void *arg)
{
        char buf[2048];
        ctftests_lookup_cb_t *clc = arg;

        if (ctf_type_name(clc->clc_fp, id, buf, sizeof (buf)) == NULL)
                return (0);

        if (strcmp(buf, clc->clc_name) != 0)
                return (0);

        clc->clc_id = id;
        return (1);
}

/*
 * This is a variant on the classic ctf_lookup_by_name(). ctf_lookup_by_name()
 * skips qualifiers, which makes sense given what the consumers of it are trying
 * to do. However, that's not what we want here. So instead we basically have to
 * walk the type table.
 */
static ctf_id_t
ctftest_lookup_type(ctf_file_t *fp, const char *name)
{
```



```
        ctftests_lookup_cb_t clc;

        clc.clc_fp = fp;
        clc.clc_id = CTF_ERR;
        clc.clc_name = name;

        (void) ctf_type_iter(fp, B_TRUE, ctftest_lookup_type_cb, &clc);
        return (clc.clc_id);
}

static int
ctftest_lookup_object_cb(const char *obj, ctf_id_t type, ulong_t idx, void *arg)
{
        ctftests_lookup_cb_t *clc = arg;

        if (strcmp(obj, clc->clc_name) == 0) {
                clc->clc_id = type;
                return (1);
        }

        return (0);
}

static ctf_id_t
ctftest_lookup_symbol(ctf_file_t *fp, const char *name)
{
        ctftests_lookup_cb_t clc;

        clc.clc_fp = fp;
        clc.clc_id = CTF_ERR;
        clc.clc_name = name;

        (void) ctf_object_iter(fp, ctftest_lookup_object_cb, &clc);
        return (clc.clc_id);
}

typedef struct ctf_function_cb {
        const char *cfc_name;
        ulong_t *cfc_symp;
        ctf_funcinfo_t *cfc_fip;
} ctf_function_cb_t;

static int
ctftest_lookup_function_cb(const char *name, ulong_t symidx,
    ctf_funcinfo_t *fip, void *arg)
{
        ctf_function_cb_t *cfc = arg;
        if (strcmp(name, cfc->cfc_name) != 0)
                return (0);

        *cfc->cfc_symp = symidx;
        *cfc->cfc_fip = *fip;

        return (1);
}

/*
 * Note, this function finds the first one with a matching name. This must not
 * be used when performing searches where a given name may occur more than once.
 */
static boolean_t
ctftest_lookup_function(ctf_file_t *fp, const char *name, ulong_t *symp,
    ctf_funcinfo_t *fip)
{
        ctf_function_cb_t cfc;
```

```
127         *symp = 0;
128         cfc.cfc_name = name;
129         cfc.cfc_symp = symp;
130         cfc.cfc_fip = fip;
131         (void) ctf_function_iter(fp, ctftest_lookup_function_cb, &cfc);
132         return (*symp == 0 ? B_FALSE : B_TRUE);
133 }

135 boolean_t
136 ctftest_check_numbers(ctf_file_t *fp, const check_number_t *tests)
137 {
138         uint_t i;
139         boolean_t ret = B_TRUE;

141         for (i = 0; tests[i].cn_tname != NULL; i++) {
142                 ctf_id_t id;
143                 ctf_encoding_t enc;

145                 id = ctftest_lookup_type(fp, tests[i].cn_tname);
146                 if (id == CTF_ERR) {
147                         warnx("failed to look up %s", tests[i].cn_tname);
148                         ret = B_FALSE;
149                         continue;
150                 }

152                 if (ctf_type_kind(fp, id) != tests[i].cn_kind) {
153                         warnx("type kind mismatch for %s: got %u, expected %u",
154                             tests[i].cn_tname, ctf_type_kind(fp, id),
155                             tests[i].cn_kind);
156                         ret = B_FALSE;
157                         continue;
158                 }

160                 if (ctf_type_encoding(fp, id, &enc) == CTF_ERR) {
161                         warnx("failed to get type encoding for %s: %s",
162                             tests[i].cn_tname, ctf_errmsg(ctf_errno(fp)));
163                         ret = B_FALSE;
164                         continue;
165                 }

167                 if (enc.cte_format != tests[i].cn_flags) {
168                         warnx("encoding flags mismatch for %s: got 0x%x, "
169                             "expected 0x%x", tests[i].cn_tname, enc.cte_format,
170                             tests[i].cn_flags);
171                         ret = B_FALSE;
172                         continue;
173                 }

175                 if (enc.cte_offset != tests[i].cn_offset) {
176                         warnx("encoding offset mismatch for %s: got 0x%x, "
177                             "expected 0x%x", tests[i].cn_tname, enc.cte_offset,
178                             tests[i].cn_offset);
179                         ret = B_FALSE;
180                         continue;
181                 }

183                 if (enc.cte_bits != tests[i].cn_size) {
184                         warnx("encoding size mismatch for %s: got 0x%x, "
185                             "expected 0x%x", tests[i].cn_tname, enc.cte_bits,
186                             tests[i].cn_size);
187                         ret = B_FALSE;
188                         continue;
189                 }
190         }

192         return (ret);
```



```
193 }

195 typedef struct ctftests_symbol_cb {
196         ctf_file_t      *csc_fp;
197         boolean_t       csc_ret;
198         const check_symbol_t *csc_tests;
199 } ctftest_symbol_cb_t;

201 static int
202 ctftest_check_symbol_cb(const char *obj, ctf_id_t type, ulong_t idx, void *arg)
203 {
204         ctftest_symbol_cb_t *cb = arg;
205         const check_symbol_t *tests = cb->csc_tests;
206         ctf_file_t *fp = cb->csc_fp;
207         uint_t i;

209         for (i = 0; tests[i].cs_symbol != NULL; i++) {
210                 ctf_id_t id;

212                 if (strcmp(obj, tests[i].cs_symbol) != 0)
213                         continue;

215                 id = ctftest_lookup_type(fp, tests[i].cs_type);
216                 if (id == CTF_ERR) {
217                         warnx("failed to lookup type %s for symbol %s",
218                             tests[i].cs_type, tests[i].cs_symbol);
219                         cb->csc_ret = B_FALSE;
220                         return (0);
221                 }

223                 if (id != type) {
224                         warnx("type mismatch for symbol %s, has type id %u, "
225                             "but specified type %s has id %u",
226                             tests[i].cs_symbol, type, tests[i].cs_type, id);
227                         cb->csc_ret = B_FALSE;
228                         return (0);
229                 }
230         }

232         return (0);
233 }

235 boolean_t
236 ctftest_check_symbols(ctf_file_t *fp, const check_symbol_t *tests)
237 {
238         ctftest_symbol_cb_t cb;

240         cb.csc_fp = fp;
241         cb.csc_ret = B_TRUE;
242         cb.csc_tests = tests;
243         if (ctf_object_iter(fp, ctftest_check_symbol_cb, &cb) != 0)
244                 return (B_FALSE);
245         return (cb.csc_ret);
246 }


249 boolean_t
250 ctftest_check_descent(const char *symbol, ctf_file_t *fp,
251     const check_descent_t *tests)
252 {
253         ctf_id_t base;
254         uint_t layer = 0;

256         /*
257          * First, find the initial type of the symbol.
258          */
```

```
259         base = ctftest_lookup_symbol(fp, symbol);
260         if (base == CTF_ERR) {
261                 warnx("failed to lookup type for symbol %s", symbol);
262                 return (B_FALSE);
263         }

265         while (tests->cd_tname != NULL) {
266                 ctf_id_t tid;
267                 int kind;
268                 ctf_arinfo_t ari;

270                 if (base == CTF_ERR) {
271                         warnx("encountered non-reference type at layer %u "
272                             "while still expecting type %s for symbol %s",
273                             layer, tests->cd_tname, symbol);
274                         return (B_FALSE);
275                 }

277                 tid = ctftest_lookup_type(fp, tests->cd_tname);
278                 if (tid == CTF_ERR) {
279                         warnx("failed to lookup type %s", tests->cd_tname);
280                         return (B_FALSE);
281                 }

283                 if (tid != base) {
284                         warnx("type mismatch at layer %u: found id %u, but "
285                             "expecting type id %u for type %s, symbol %s",
286                             layer, base, tid, tests->cd_tname, symbol);
287                         return (B_FALSE);
288                 }

290                 kind = ctf_type_kind(fp, base);
291                 if (kind != tests->cd_kind) {
292                         warnx("type kind mismatch at layer %u: found kind %u, "
293                             "but expected kind %u for %s, symbol %s", layer,
294                             kind, tests->cd_kind, tests->cd_tname, symbol);
295                         return (B_FALSE);
296                 }

298                 switch (kind) {
299                 case CTF_K_ARRAY:
300                         if (ctf_array_info(fp, base, &ari) == CTF_ERR) {
301                                 warnx("failed to lookup array info at layer "
302                                     "%u for type %s, symbol %s: %s", base,
303                                     tests->cd_tname, symbol,
304                                     ctf_errmsg(ctf_errno(fp)));
305                                 return (B_FALSE);
306                         }

308                         if (tests->cd_nents != ari.ctr_nelems) {
309                                 warnx("array element mismatch at layer %u "
310                                     "for type %s, symbol %s: found %u, "
311                                     "expected %u", layer, tests->cd_tname,
312                                     symbol, ari.ctr_nelems, tests->cd_nents);
313                                 return (B_FALSE);
314                         }

316                         tid = ctftest_lookup_type(fp, tests->cd_contents);
317                         if (tid == CTF_ERR) {
318                                 warnx("failed to look up type %s",
319                                     tests->cd_contents);
320                                 return (B_FALSE);
321                         }

323                         if (ari.ctr_contents != tid) {
324                                 warnx("array contents mismatch at layer %u "
```



```
325                                     "for type %s, symbol %s: found %u, "
326                                     "expected %s/%u", layer, tests->cd_tname,
327                                     symbol, ari.ctr_contents,
328                                     tests->cd_contents, tid);

330                                 return (B_FALSE);
331                         }
332                         base = ari.ctr_contents;
333                         break;
334                 default:
335                         base = ctf_type_reference(fp, base);
336                         break;
337                 }

339                 tests++;
340                 layer++;
341         }

343         if (base != CTF_ERR) {
344                 warnx("found additional type %u in chain, but expected no more",
345                     base);
346                 return (B_FALSE);
347         }

349         return (B_TRUE);
350 }

352 int
353 ctftest_check_enum_count(const char *name, int value, void *arg)
354 {
355         uint_t *u = arg;
356         *u = *u + 1;
357         return (0);
358 }

360 int
361 ctftest_check_enum_value(const char *name, int value, void *arg)
362 {
363         uint_t i;
364         const check_enum_t *enums = arg;

366         for (i = 0; enums[i].ce_name != NULL; i++) {
367                 if (strcmp(enums[i].ce_name, name) != 0)
368                         continue;
369                 if (enums[i].ce_value == (int64_t)value)
370                         return (0);
371                 warnx("enum %s value mismatch: found %d, expected %" PRId64,
372                     name, value, enums[i].ce_value);
373                 return (1);
374         }

376         warnx("found no matching entry for enum member %s", name);
377         return (1);
378 }

380 boolean_t
381 ctftest_check_enum(const char *type, ctf_file_t *fp, const check_enum_t *enums)
382 {
383         int ret;
384         uint_t tcount, ecount;
385         ctf_id_t base;

387         if ((base = ctftest_lookup_type(fp, type)) == CTF_ERR) {
388                 warnx("Failed to look up type %s", type);
389                 return (B_FALSE);
390         }
```

```
392         if (ctf_type_kind(fp, base) != CTF_K_ENUM) {
393                 warnx("%s is not an enum", type);
394                 return (B_FALSE);
395         }
396
397         /*
398          * First count how many entries we have.
399          */
400         tcount = 0;
401         while (enums[tcount].ce_name != NULL) {
402                 tcount++;
403         }
404
405         ecount = 0;
406         if (ctf_enum_iter(fp, base, ctftest_check_enum_count, &ecount) != 0) {
407                 warnx("failed to walk enum %s: %s", type,
408                     ctf_errmsg(ctf_errno(fp)));
409                 return (B_FALSE);
410         }
411
412         if (tcount != ecount) {
413                 warnx("enum value mismatch: expected %u values, but found %u",
414                     tcount, ecount);
415                 return (B_FALSE);
416         }
417
418         if ((ret = ctf_enum_iter(fp, base, ctftest_check_enum_value,
419             (void *)enums)) != 0) {
420                 if (ret == -1) {
421                         warnx("failed to walk enum %s: %s", type,
422                             ctf_errmsg(ctf_errno(fp)));
423                 }
424                 return (B_FALSE);
425         }
426
427         return (B_TRUE);
428 }
429
430 int
431 ctftest_check_member_count(const char *mname, ctf_id_t mtype, ulong_t bitoff,
432     void *arg)
433 {
434         uint_t *countp = arg;
435         *countp = *countp + 1;
436         return (0);
437 }
438
439 int
440 ctftest_check_members_cb(const char *mname, ctf_id_t mtype, ulong_t bitoff,
441     void *arg)
442 {
443         uint_t i;
444         const ctftest_member_cb_t *cmc = arg;
445         const check_member_t *members = cmc->cmc_members;
446         ctf_file_t *fp = cmc->cmc_fp;
447
448         for (i = 0; members[i].cm_name != NULL; i++) {
449                 boolean_t bad = B_FALSE;
450                 char buf[2048];
451
452                 if (strcmp(mname, members[i].cm_name) != 0)
453                         continue;
454
455                 if (bitoff != members[i].cm_offset) {
456                         warnx("member %s of type %s has mismatched bit offset: "
```



```
457                             "found %lu, expected %lu", mname, cmc->cmc_name,
458                             bitoff, members[i].cm_offset);
459                         bad = B_TRUE;
460                 }
461
462                 if (ctf_type_name(fp, mtype, buf, sizeof (buf)) == NULL) {
463                         warnx("failed to obtain type name for member %s",
464                             mname, ctf_errmsg(ctf_errno(fp)));
465                         bad = B_TRUE;
466                 } else if (strcmp(buf, members[i].cm_type) != 0) {
467                         warnx("member %s has bad type, found %s, expected %s",
468                             mname, buf, members[i].cm_type);
469                         bad = B_TRUE;
470                 }
471
472                 return (bad ? 1 : 0);
473         }
474
475         warnx("found no matching entry for member %s of type %s", mname,
476             cmc->cmc_name);
477         return (1);
478 }
479
480 boolean_t
481 ctftest_check_members(const char *type, ctf_file_t *fp, int kind,
482     size_t size, const check_member_t *members)
483 {
484         int ret;
485         uint_t tcount, mcount;
486         ctf_id_t base;
487         ctftest_member_cb_t cmc;
488
489         if ((base = ctftest_lookup_type(fp, type)) == CTF_ERR) {
490                 warnx("failed to look up type %s", type);
491                 return (B_FALSE);
492         }
493
494         if (ctf_type_kind(fp, base) != kind) {
495                 warnx("%s has kind %s, expected %s", type,
496                     ctf_kind_name(fp, ctf_type_kind(fp, base)),
497                     ctf_kind_name(fp, kind));
498                 return (B_FALSE);
499         }
500
501         if (size != ctf_type_size(fp, base)) {
502                 warnx("%s has bad size, expected %lu, found %lu",
503                     type, size, ctf_type_size(fp, base));
504                 return (B_FALSE);
505         }
506
507         /*
508          * First count how many entries we have.
509          */
510         tcount = 0;
511         while (members[tcount].cm_name != NULL) {
512                 tcount++;
513         }
514
515         mcount = 0;
516         if (ctf_member_iter(fp, base, ctftest_check_member_count, &mcount) !=
517             0) {
518                 warnx("failed to walk members of %s: %s", type,
519                     ctf_errmsg(ctf_errno(fp)));
520                 return (B_FALSE);
521         }
```

```
523         if (tcount != mcount) {
524                 warnx("type member mismatch: expected %u values, but found %u",
525                     tcount, mcount);
526                 return (B_FALSE);
527         }

529         cmc.cmc_fp = fp;
530         cmc.cmc_members = members;
531         cmc.cmc_name = type;
532         if ((ret = ctf_member_iter(fp, base, ctftest_check_members_cb,
533             &cmc)) != 0) {
534                 if (ret == -1) {
535                         warnx("failed to walk type %s: %s", type,
536                             ctf_errmsg(ctf_errno(fp)));
537                 }
538                 return (B_FALSE);
539         }

541         return (B_TRUE);
542 }

544 boolean_t
545 ctftest_check_function(const char *symbol, ctf_file_t *fp, const char *rtype,
546     uint_t nargs, uint_t flags, const char **argv)
547 {
548         ulong_t sym;
549         ctf_funcinfo_t fi;
550         uint_t i;
551         boolean_t ret = B_TRUE;
552         ctf_id_t *args;
553         char buf[2048];


556         if (!ctftest_lookup_function(fp, symbol, &sym, &fi)) {
557                 warnx("failed to look up function %s", symbol);
558                 return (B_FALSE);
559         }

561         if (ctf_type_name(fp, fi.ctc_return, buf, sizeof (buf)) == NULL) {
562                 warnx("failed to lookup return type name for function %s",
563                     symbol);
564                 ret = B_FALSE;
565         } else if (strcmp(rtype, buf) != 0) {
566                 warnx("return type has wrong type: found %s, expected %s",
567                     buf, rtype);
568                 ret = B_FALSE;
569         }

571         if (nargs != fi.ctc_argc) {
572                 warnx("function argument mismatch: found %u, expected %u",
573                     fi.ctc_argc, nargs);
574                 ret = B_FALSE;
575         }

577         if (flags != fi.ctc_flags) {
578                 warnx("function flags mismatch, found 0x%x, expected 0x%x",
579                     fi.ctc_flags, flags);
580                 ret = B_FALSE;
581         }

583         if (!ret || fi.ctc_argc == 0) {
584                 return (ret);
585         }

587         if ((args = calloc(fi.ctc_argc, sizeof (ctf_id_t))) == NULL) {
588                 warnx("failed to allocate memory for function arguments");
```



```
589                 return (B_FALSE);
590         }

592         if (ctf_func_args(fp, sym, fi.ctc_argc, args) != 0) {
593                 warnx("failed to get function information: %s",
594                     ctf_errmsg(ctf_errno(fp)));
595                 free(args);
596                 return (B_FALSE);
597         }

599         for (i = 0; i < fi.ctc_argc; i++) {
600                 if (ctf_type_name(fp, args[i], buf, sizeof (buf)) == NULL) {
601                         warnx("failed to obtain type name for argument %u",
602                             i, ctf_errmsg(ctf_errno(fp)));
603                         ret = B_FALSE;
604                         break;
605                 }

607                 if (strcmp(buf, argv[i]) != 0) {
608                         warnx("argument %u has wrong type: found %s, "
609                             "expected %s", i, buf, argv[i]);
610                         ret = B_FALSE;
611                         break;
612                 }
613         }

615         free(args);
616         return (ret);
617 }

619 boolean_t
620 ctftest_check_fptr(const char *type, ctf_file_t *fp, const char *rtype,
621     uint_t nargs, uint_t flags, const char **argv)
622 {
623         ctf_id_t tid;
624         ctf_funcinfo_t fi;
625         uint_t i;
626         boolean_t ret = B_TRUE;
627         ctf_id_t *args;
628         char buf[2048];


631         if ((tid = ctf_lookup_by_name(fp, type)) == CTF_ERR) {
632                 warnx("failed to look up type %s: %s", type,
633                     ctf_errmsg(ctf_errno(fp)));
634                 return (B_FALSE);
635         }

637         /*
638          * Perform two CTF type resolves, one for the function pointer and one
639          * for the typedef that gets passed in.
640          */
641         if ((tid = ctf_type_resolve(fp, tid)) == CTF_ERR) {
642                 warnx("failed to convert type %s to base type: %s", type,
643                     ctf_errmsg(ctf_errno(fp)));
644                 return (B_FALSE);
645         }

647         if (ctf_type_kind(fp, tid) == CTF_K_POINTER &&
648             (tid = ctf_type_reference(fp, tid)) == CTF_ERR) {
649                 warnx("failed to convert type %s to base type: %s", type,
650                     ctf_errmsg(ctf_errno(fp)));
651                 return (B_FALSE);
652         }

654         if (ctf_func_info_by_id(fp, tid, &fi) != 0) {
```

```
655                 warnx("failed to get function information for type %s: %s",
656                     type, ctf_errmsg(ctf_errno(fp)));
657                 return (B_FALSE);
658         }

660         if (ctf_type_name(fp, fi.ctc_return, buf, sizeof (buf)) == NULL) {
661                 warnx("failed to lookup return type name for function %s",
662                     type);
663                 ret = B_FALSE;
664         } else if (strcmp(rtype, buf) != 0) {
665                 warnx("return type has wrong type: found %s, expected %s",
666                     buf, rtype);
667                 ret = B_FALSE;
668         }

670         if (nargs != fi.ctc_argc) {
671                 warnx("function argument mismatch: found %u, expected %u",
672                     fi.ctc_argc, nargs);
673                 ret = B_FALSE;
674         }

676         if (flags != fi.ctc_flags) {
677                 warnx("function flags mismatch, found 0x%x, expected 0x%x",
678                     fi.ctc_flags, flags);
679                 ret = B_FALSE;
680         }

682         if (!ret || fi.ctc_argc == 0) {
683                 return (ret);
684         }

686         if ((args = calloc(fi.ctc_argc, sizeof (ctf_id_t))) == NULL) {
687                 warnx("failed to allocate memory for function arguments");
688                 return (B_FALSE);
689         }

691         if (ctf_func_args_by_id(fp, tid, fi.ctc_argc, args) != 0) {
692                 warnx("failed to get function information: %s",
693                     ctf_errmsg(ctf_errno(fp)));
694                 free(args);
695                 return (B_FALSE);
696         }

698         for (i = 0; i < fi.ctc_argc; i++) {
699                 if (ctf_type_name(fp, args[i], buf, sizeof (buf)) == NULL) {
700                         warnx("failed to obtain type name for argument %u",
701                             i, ctf_errmsg(ctf_errno(fp)));
702                         ret = B_FALSE;
703                         break;
704                 }

706                 if (strcmp(buf, argv[i]) != 0) {
707                         warnx("argument %u has wrong type: found %s, "
708                             "expected %s", i, buf, argv[i]);
709                         ret = B_FALSE;
710                         break;
711                 }
712         }

714         free(args);
715         return (ret);
716 }

718 typedef struct ctftest_duplicates {
719         ctf_file_t *ctd_fp;
720         char **ctd_names;
```



```
721         size_t ctd_len;
722         size_t ctd_curent;
723         boolean_t ctd_ret;
724 } ctftest_duplicates_t;

726 static int
727 ctftest_duplicates_cb(ctf_id_t id, boolean_t root, void *arg)
728 {
729         char buf[2048];
730         ctftest_duplicates_t *dup = arg;
731         size_t i;

733         if (ctf_type_name(dup->ctd_fp, id, buf, sizeof (buf)) == NULL) {
734                 warnx("failed to lookup name for id %ld", id);
735                 dup->ctd_ret = B_FALSE;
736                 return (1);
737         }

739         for (i = 0; i < dup->ctd_curent; i++) {
740                 if (strcmp(buf, dup->ctd_names[i]) == 0) {
741                         warnx("encountered duplicate type '%s'", buf);
742                         dup->ctd_ret = B_FALSE;
743                         /*
744                          * Don't break out of the loop and keep going in case we
745                          * find another duplicate.
746                          */
747                         return (0);
748                 }
749         }

751         if (dup->ctd_curent == dup->ctd_len) {
752                 char **n;
753                 size_t newlen = dup->ctd_len * 2;

755                 n = recallocarray(dup->ctd_names, dup->ctd_len, newlen,
756                     sizeof (char *));
757                 if (n == NULL) {
758                         warnx("failed to resize type name array");
759                         dup->ctd_ret = B_FALSE;
760                         return (1);
761                 }

763                 dup->ctd_names = n;
764                 dup->ctd_len = newlen;
765         }

767         dup->ctd_names[dup->ctd_curent] = strdup(buf);
768         if (dup->ctd_names[dup->ctd_curent] == NULL) {
769                 warn("failed to duplicate type name");
770                 dup->ctd_ret = B_FALSE;
771                 return (1);
772         }
773         dup->ctd_curent++;

775         return (0);
776 }

778 boolean_t
779 ctftest_duplicates(ctf_file_t *fp)
780 {
781         size_t i;
782         ctftest_duplicates_t d;

784         bzero(&d, sizeof (d));
785         d.ctd_fp = fp;
786         d.ctd_len = 4;
```

```
787         d.ctd_ret = B_TRUE;
788         d.ctd_names = recallocarray(NULL, 0, d.ctd_len, sizeof (char *));
789         if (d.ctd_names == NULL) {
790                 warnx("failed to allocate duplicate name storage");
791                 return (B_FALSE);
792         }

794         (void) ctf_type_iter(fp, B_TRUE, ctftest_duplicates_cb, &d);

796         for (i = 0; i < d.ctd_curent; i++) {
797                 free(d.ctd_names[i]);
798         }
799         free(d.ctd_names);

801         return (d.ctd_ret);
802 }
```

```
**********************************************************
    3441 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-common.h
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#ifndef _CHECK_COMMON_H
#define _CHECK_COMMON_H

/*
 * Common definitions for the CTF tests
 */

#include <stdlib.h>
#include <unistd.h>
#include <libctf.h>
#include <err.h>
#include <strings.h>
#include <sys/param.h>

#ifdef __cplusplus
extern "C" {
#endif

typedef struct check_number {
        const char *cn_tname;
        uint_t cn_kind;
        uint_t cn_flags;
        uint_t cn_offset;
        uint_t cn_size;
} check_number_t;

typedef struct check_symbol {
        const char *cs_symbol;
        const char *cs_type;
} check_symbol_t;

typedef struct check_descent {
        const char *cd_tname;
        uint_t cd_kind;
        const char *cd_contents;
        uint_t cd_nents;
} check_descent_t;

typedef struct check_descent_test {
        const char *cdt_sym;
        const check_descent_t *cdt_tests;
} check_descent_test_t;

typedef struct check_enum {
        const char *ce_name;
```



```
        int64_t ce_value;
} check_enum_t;

typedef struct check_enum_test {
        const char *cet_type;
        const check_enum_t *cet_tests;
} check_enum_test_t;

typedef struct check_member {
        const char *cm_name;
        const char *cm_type;
        ulong_t cm_offset;
} check_member_t;

typedef struct check_member_test {
        const char *cmt_type;
        int cmt_kind;
        size_t cmt_size;
        const check_member_t *cmt_members;
} check_member_test_t;

typedef struct check_function_test {
        const char *cft_name;
        const char *cft_rtype;
        uint_t cft_nargs;
        uint_t cft_flags;
        const char **cft_args;
} check_function_test_t;

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

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

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

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

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

/*
 * Check that the named function or function pointer has the correct return
 * type, arguments, and function flags.
 */
```

```
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 }
139 #endif

141 #endif /* _CHECK_COMMON_H */
```

```
**********************************************************
    2869 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-enum.c
10814 Want primordial CTF test suite
Reviewed 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) 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 }
};

static check_descent_t check_descent_ff6[] = {
        { "enum ff6", CTF_K_ENUM },
        { NULL }
};

static check_descent_t check_descent_ff10[] = {
        { "ff10_t", CTF_K_TYPEDEF },
        { "enum ff10", CTF_K_ENUM },
        { NULL }
};

static check_descent_t check_descent_chrono[] = {
        { "chrono_t", CTF_K_TYPEDEF },
        { "enum chrono", CTF_K_ENUM },
        { NULL }
};

static check_descent_test_t descents[] = {
        { "ff10", check_descent_ff10 },
        { "ff6", check_descent_ff6 },
        { "trigger", check_descent_chrono },
        { NULL }
};

static check_enum_t check_enum_ff6[] = {
        { "TERRA", 0 },
        { "LOCKE", 1 },
        { "EDGAR", 2 },
        { "SABIN", 3 },
        { "CELES", 4 },
        { "CYAN", 5 },
        { "SHADOW", 6 },
        { "GAU", 7 },
```



```
        { "SETZER", 8 },
        { "STRAGO", 9 },
        { "RELM", 10 },
        { "MOG", 11 },
        { "GOGO", 12 },
        { "UMARO", 13 },
        { "LEO", 14 },
        { "KEFKA", 15 },
        { NULL }
};

static check_enum_t check_enum_ff10[] = {
        { "TIDUS", -10 },
        { "YUNA", 23 },
        { "AURON", -34 },
        { "WAKA", 52 },
        { "LULU", INT32_MAX },
        { "RIKKU", INT32_MIN },
        { "KHIMARI", 0 },
        { NULL }
};

static check_enum_t check_enum_chrono[] = {
        { "CRONO", 0x1000 },
        { "LUCCA", 0x2000 },
        { "MARLE", 0x3000 },
        { "FROG", 0x4000 },
        { "ROBO", 0x5000 },
        { "AYLA", 0x6000 },
        { "MAGUS", 0x7000 },
        { "SCHALA", 0x8000 },
        { "LAVOS", 0x9000 },
        { "BALTHAZAR", 0xa000 },
        { NULL }
};

static check_enum_test_t enums[] = {
        { "enum ff6", check_enum_ff6 },
        { "enum ff10", check_enum_ff10 },
        { "enum chrono", check_enum_chrono },
        { 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;
                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,
```

```
127                             descents[d].cdt_tests)) {
128                                 ret = EXIT_FAILURE;
129                         }
130                 }

132                 for (d = 0; enums[d].cet_type != NULL; d++) {
133                         if (!ctftest_check_enum(enums[d].cet_type, fp,
134                             enums[d].cet_tests)) {
135                                 ret = EXIT_FAILURE;
136                         }
137                 }
138                 ctf_close(fp);
139         }

141         return (ret);

143 }
```

```
*********************************************************
    1845 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-float.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Check for basic float types.
 */

#include <stdlib.h>
#include <unistd.h>

#include "check-common.h"

static check_number_t check_floats[] = {
        { "float", CTF_K_FLOAT, CTF_FP_SINGLE, 0, 32 },
        { "double", CTF_K_FLOAT, CTF_FP_DOUBLE, 0, 64 },
#ifdef  TARGET_LP64
        { "long double", CTF_K_FLOAT, CTF_FP_LDOUBLE, 0, 128 },
#else
        { "long double", CTF_K_FLOAT, CTF_FP_LDOUBLE, 0, 96 },
#endif
        { "complex float", CTF_K_FLOAT, CTF_FP_CPLX, 0, 64 },
        { "complex double", CTF_K_FLOAT, CTF_FP_DCPLX, 0, 128 },
#ifdef  TARGET_LP64
        { "complex long double", CTF_K_FLOAT, CTF_FP_LDCPLX, 0, 256 },
#else
        { "complex long double", CTF_K_FLOAT, CTF_FP_LDCPLX, 0, 192 },
#endif
        { NULL }
};

static check_symbol_t check_syms[] = {
        { "a", "float" },
        { "b", "double" },
        { "c", "long double" },
        { "d", "complex float" },
        { "e", "complex double" },
        { "f", "complex long double" },
        { 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;

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

        return (ret);
}
```

```
**********************************************************
    3196 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-forward.c
10814 Want primordial CTF test suite
Reviewed 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) 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 }
};

static check_member_t check_member_forward[] = {
#ifdef  TARGET_LP64
        { "prev", "struct foo *", 0 },
        { "next", "struct foo *", 8 * NBBY },
        { "data", "struct bar *", 16 * NBBY },
        { "tag", "enum baz *", 24 * NBBY },
#else
        { "prev", "struct foo *", 0 },
        { "next", "struct foo *", 4 * NBBY },
        { "data", "struct bar *", 8 * NBBY },
        { "tag", "enum baz *", 12 * NBBY },
#endif
        { NULL }
};


static check_member_test_t members[] = {
#ifdef  TARGET_LP64
        { "struct forward", CTF_K_STRUCT, 32, check_member_forward },
#else
        { "struct forward", CTF_K_STRUCT, 16, check_member_forward },
#endif
        { NULL }
```



```
};

static check_descent_t check_descent_foo[] = {
        { "struct foo *", CTF_K_POINTER },
        { "struct foo", CTF_K_FORWARD },
        { NULL }
};

static check_descent_t check_descent_bar[] = {
        { "struct bar *", CTF_K_POINTER },
        { "struct bar", CTF_K_FORWARD },
        { NULL }
};

static check_descent_t check_descent_baz[] = {
        { "enum baz *", CTF_K_POINTER },
        { "enum baz", CTF_K_ENUM },
        { NULL }
};

static check_descent_test_t descents[] = {
        { "foop", check_descent_foo },
        { "barp", check_descent_bar },
        { "bazp", check_descent_baz },
        { 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;
                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)) {
                                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);
```

```
127         }

129         return (ret);
130 }
```

```
**********************************************************
    2358 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-function.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Check that we properly handle functions and function pointers.
 */

#include "check-common.h"

static const char *one_args[] = { "int" };
static const char *two_args[] = { "int", "const char *" };
static const char *three_args[] = { "int", "const char *", "float" };
static const char *argument_args[] = { "uintptr_t" };
static const char *vararg_args[] = { "const char *" };

static check_function_test_t functions[] = {
        { "simple_func", "void", 0, 0, NULL },
        { "one", "void", 1, 0, one_args },
        { "two", "void", 2, 0, two_args },
        { "three", "void", 3, 0, three_args },
        { "noarg", "const char *", 0, 0, NULL },
        { "argument", "const char *", 1, 0, argument_args },
        { "vararg", "void", 1, CTF_FUNC_VARARG, vararg_args },
        { "vararg_ret", "uintptr_t", 1, CTF_FUNC_VARARG, vararg_args },
        { NULL }
};

static const char *strfunc_args[] = { "const char *", "const char *" };

static check_function_test_t fptrs[] = {
        { "strfunc_t", "int", 2, 0, strfunc_args },
        { "vararg_t", "void", 1, CTF_FUNC_VARARG, vararg_args },
        { 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;
                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;
                }

                for (j = 0; functions[j].cft_name != NULL; j++) {
                        if (!ctftest_check_function(functions[j].cft_name, fp,
                            functions[j].cft_rtype, functions[j].cft_nargs,
                            functions[j].cft_flags, functions[j].cft_args)) {
                                ret = EXIT_FAILURE;
                        }
                }

                for (j = 0; fptrs[j].cft_name != NULL; j++) {
                        if (!ctftest_check_fptr(fptrs[j].cft_name, fp,
                            fptrs[j].cft_rtype, fptrs[j].cft_nargs,
                            fptrs[j].cft_flags, fptrs[j].cft_args)) {
                                ret = EXIT_FAILURE;
                        }
                }

                ctf_close(fp);
        }

        return (ret);
}
```

```
**********************************************************
    2096 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-int.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Check for basic integer types.
 */

#include <stdlib.h>
#include <unistd.h>

#include "check-common.h"

static check_number_t check_ints[] = {
        { "char", CTF_K_INTEGER, CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 },
        { "short", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 16 },
        { "int", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 32 },
#ifdef  TARGET_LP64
        { "long", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 64 },
#else
        { "long", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 32 },
#endif
        { "long long", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 64 },
        { "unsigned char", CTF_K_INTEGER, CTF_INT_CHAR, 0, 8 },
        { "unsigned short", CTF_K_INTEGER, 0, 0, 16 },
        { "unsigned int", CTF_K_INTEGER, 0, 0, 32 },
#ifdef  TARGET_LP64
        { "unsigned long", CTF_K_INTEGER, 0, 0, 64 },
#else
        { "unsigned long", CTF_K_INTEGER, 0, 0, 32 },
#endif
        { "unsigned long long", CTF_K_INTEGER, 0, 0, 64 },
        { NULL }
};

static check_symbol_t check_syms[] = {
        { "a", "char" },
        { "b", "unsigned char" },
        { "d", "short" },
        { "e", "unsigned short" },
        { "g", "int" },
        { "h", "unsigned int" },
        { "j", "long" },
        { "k", "unsigned long" },
        { "m", "long long" },
        { "n", "unsigned long long" },
        { 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;

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

        return (ret);
}
```

```
**********************************************************
    1626 Tue Apr 23 05:35:49 2019
new/usr/src/test/util-tests/tests/ctf/check-merge-dedup.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * This tests that we don't end up with several copies of the same type.
 */

#include "check-common.h"

static check_symbol_t check_syms[] = {
        { "int", "a" },
        { "short", "b" },
        { "const char *", "c" },
        { "float", "d" },
        { "double" "e" },
        { "int", "f" },
        { "short", "g" },
        { "const char *", "h" },
        { "float", "i" },
        { "double" "j" },
        { "int", "k" },
        { "short", "l" },
        { "const char *", "m" },
        { "float", "n" },
        { "double" "o" },
        { "int", "p" },
        { "short", "q" },
        { "const char *", "r" },
        { "float", "s" },
        { "double" "t" },
        { "struct dup" "dupmain" },
        { "struct dup" "dup1" },
        { "struct dup" "dup2" },
        { "struct dup" "dup3" },
        { 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;

                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;
                }

                if (!ctftest_duplicates(fp)) {
                        ret = EXIT_FAILURE;
                }

                ctf_close(fp);
        }

        return (ret);
}
```

```
**********************************************************
    2975 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/check-merge-forward.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * This tests that a forward declared in one object file that is defined in
 * another doesn't end up in the final one.
 */

#include "check-common.h"

static check_symbol_t check_syms[] = {
        { "list", "foo_list_t" },
        { NULL }
};

static check_member_t check_member_foo_list[] = {
        { "count", "int", 0 },
#ifdef  TARGET_LP64
        { "head", "struct foo *", 8 * NBBY },
        { "tail", "struct foo *", 16 * NBBY },
#else
        { "head", "struct foo *", 4 * NBBY },
        { "tail", "struct foo *", 8 * NBBY },
#endif
        { NULL }
};

static check_member_t check_member_foo[] = {
        { "next", "struct foo *", 0 * NBBY },
#ifdef  TARGET_LP64
        { "left", "int", 8 * NBBY },
        { "right", "int", 12 * NBBY },
        { "count", "int", 16 * NBBY },
#else
        { "left", "int", 4 * NBBY },
        { "right", "int", 8 * NBBY },
        { "count", "int", 12 * NBBY },
#endif
        { NULL }
};

static check_member_test_t members[] = {
#ifdef  TARGET_LP64
        { "struct foo_list", CTF_K_STRUCT, 24, check_member_foo_list },
        { "struct foo", CTF_K_STRUCT, 24, check_member_foo },
#else
        { "struct foo_list", CTF_K_STRUCT, 12, check_member_foo_list },
        { "struct foo", CTF_K_STRUCT, 16, check_member_foo },
```



```
#endif
        { NULL }
};

static int
ctf_merge_forward_cb(ctf_id_t id, boolean_t root, void *arg)
{
        ctf_file_t *fp = arg;
        char buf[2048];

        if (ctf_type_kind(fp, id) != CTF_K_FORWARD)
                return (0);

        if (ctf_type_name(fp, id, buf, sizeof (buf)) == NULL) {
                warnx("failed to lookup the name of type %ld: %s", id,
                    ctf_errmsg(ctf_errno(fp)));
                return (1);
        }

        /*
         * If a forward shows up, that's OK. It's only bad if it's the name of
         * the one we created.
         */
        if (strcmp("struct foo", buf) == 0) {
                warnx("encountered forward type for struct foo that "
                    "shouldn't exist");
                return (1);
        }

        return (0);
}

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; 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;
                        }
                }

                if (ctf_type_iter(fp, B_TRUE, ctf_merge_forward_cb, fp) != 0) {
                        ret = EXIT_FAILURE;
                }
```

```
128                 ctf_close(fp);
129         }

131         return (ret);
132 }
```

```
**********************************************************
    1581 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/check-merge-reduction.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * This tests that a global that has been scoped to local scope through symbol
 * reduction of a mapfile can still be detected.
 */

#include "check-common.h"

static check_symbol_t check_syms[] = {
        { "data", "int" },
        { NULL }
};

static const char *scoped_args[] = { "uint32_t" };

static check_function_test_t functions[] = {
        { "global", "int", 0, 0, NULL },
        { "scoped", "int", 1, 0, scoped_args },
        { 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;
                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; functions[j].cft_name != NULL; j++) {
                        if (!ctftest_check_function(functions[j].cft_name, fp,
```



```
                            functions[j].cft_rtype, functions[j].cft_nargs,
                            functions[j].cft_flags, functions[j].cft_args)) {
                                ret = EXIT_FAILURE;
                        }
                }

                ctf_close(fp);
        }

        return (ret);
}
```

```
**********************************************************
    6837 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/check-merge-static.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Verify that various type information for static symbols is accurate for the
 * file in question. To run this test, there's a global and static version of a
 * symbol and function that exists on a per-file basis. These will all have been
 * reproduced in the output file. As such, we need to iterate the symbol table
 * to know which version should be which and use that to drive things.
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <libelf.h>
#include <gelf.h>
#include <limits.h>
#include <strings.h>

#include "check-common.h"

typedef struct check_map {
        const char *map_file;
        const char *map_type;
} check_map_t;

static const char *global_type = "int";
static check_map_t map[] = {
        { "test-a.c", "uint8_t" },
        { "test-b.c", "uint16_t" },
        { "test-c.c", "uint32_t" },
        { "test-d.c", "uint64_t" },
        { NULL }
};

static const char *
check_file_to_type(GElf_Sym *symp, const char *file, const char *name)
{
        uint_t i;

        if (ELF32_ST_BIND(symp->st_info) == STB_GLOBAL) {
                return (global_type);
        }

        if (file == NULL) {
                warnx("encountered non-global symbol without STT_FILE info: %s",
                    name);
                return (NULL);
        }
```



```
        }

        for (i = 0; map[i].map_file != NULL; i++) {
                if (strcmp(map[i].map_file, file) == 0) {
                        return (map[i].map_type);
                }
        }

        warnx("failed to find type mapping for symbol %s, file %s", name, file);
        return (NULL);
}

static int
check_global(ctf_file_t *fp, GElf_Sym *symp, int symid, const char *file,
    const char *name)
{
        const char *type;
        ctf_id_t tid;
        char buf[2048];

        if ((type = check_file_to_type(symp, file, name)) == NULL) {
                return (EXIT_FAILURE);
        }

        if ((tid = ctf_lookup_by_symbol(fp, symid)) == CTF_ERR) {
                warnx("failed to get type for symbol %s (%d): %s", name, symid,
                    ctf_errmsg(ctf_errno(fp)));
                return (EXIT_FAILURE);
        }

        if (ctf_type_name(fp, tid, buf, sizeof (buf)) == NULL) {
                warnx("failed to get type name for symbol %s (%d): %s",
                    name, symid, ctf_errmsg(ctf_errno(fp)));
                return (EXIT_FAILURE);
        }

        if (strcmp(buf, type) != 0) {
                warnx("type mismatch for symbol %s (%d): found %s, expected %s",
                    name, symid, buf, type);
                return (EXIT_FAILURE);
        }

        return (0);
}

static int
check_mumble(ctf_file_t *fp, GElf_Sym *symp, int symid, const char *file,
    const char *name)
{
        const char *type;
        ctf_funcinfo_t fi;
        ctf_id_t id, args;

        if ((type = check_file_to_type(symp, file, name)) == NULL) {
                return (EXIT_FAILURE);
        }

        if ((id = ctf_lookup_by_name(fp, type)) == CTF_ERR) {
                warnx("failed to lookup type id for %s: %s", type,
                    ctf_errmsg(ctf_errno(fp)));
                return (EXIT_FAILURE);
        }

        if (ctf_func_info(fp, symid, &fi) != 0) {
                warnx("failed to get function information for %s (%d): %s",
                    name, symid, ctf_errmsg(ctf_errno(fp)));
```

```
127                 return (EXIT_FAILURE);
128         }

130         if (fi.ctc_argc != 1) {
131                 warnx("argument count mismatch for symbol %s (%d): found %u, "
132                     "expected %d", name, symid, fi.ctc_argc, 1);
133                 return (EXIT_FAILURE);
134         }

136         if (fi.ctc_flags != 0) {
137                 warnx("function flags mismatch for symbol %s (%d): found %u, "
138                     "expected %d", name, symid, fi.ctc_flags, 0);
139                 return (EXIT_FAILURE);
140         }

142         if (fi.ctc_return != id) {
143                 warnx("return value mismatch for symbol %s (%d): found %ld, "
144                     "expected %ld", name, symid, fi.ctc_return, id);
145                 return (EXIT_FAILURE);
146         }

148         if (ctf_func_args(fp, symid, 1, &args) != 0) {
149                 warnx("failed to get function arguments for symbol %s (%d): %s",
150                     name, symid, ctf_errmsg(ctf_errno(fp)));
151                 return (EXIT_FAILURE);
152         }

154         if (args != id) {
155                 warnx("argument mismatch for symbol %s (%d): found %ld, "
156                     "expected %ld", name, symid, args, id);
157                 return (EXIT_FAILURE);
158         }

160         return (0);
161 }

163 static int
164 check_merge_static(const char *file, ctf_file_t *fp, Elf *elf)
165 {
166         Elf_Scn *scn = NULL, *symscn = NULL;
167         Elf_Data *symdata = NULL;
168         GElf_Shdr symhdr;
169         ulong_t nsyms;
170         int i;
171         const char *curfile = NULL;
172         int ret = 0;

174         while ((scn = elf_nextscn(elf, scn)) != NULL) {
175                 if (gelf_getshdr(scn, &symhdr) == NULL) {
176                         warnx("failed to get section header: %s",
177                             elf_errmsg(elf_errno()));
178                         return (EXIT_FAILURE);
179                 }

181                 if (symhdr.sh_type == SHT_SYMTAB) {
182                         symscn = scn;
183                         break;
184                 }
185         }

187         if (symscn == NULL) {
188                 warnx("failed to find symbol table for %s", file);
189                 return (EXIT_FAILURE);
190         }

192         if ((symdata = elf_getdata(symscn, NULL)) == NULL) {
```



```
193                 warnx("failed to get data for symbol table %s: %s", file,
194                     elf_errmsg(elf_errno()));
195                 return (EXIT_FAILURE);
196         }

198         if (symhdr.sh_link == SHN_XINDEX) {
199                 warnx("test does not support extended ELF sections!");
200                 return (EXIT_FAILURE);
201         }
202         nsyms = symhdr.sh_size / symhdr.sh_entsize;
203         if (nsyms > INT_MAX) {
204                 warnx("file contains more symbols than libelf can iterate");
205                 return (EXIT_FAILURE);
206         }

208         for (i = 1; i < (int)nsyms; i++) {
209                 GElf_Sym sym;
210                 const char *name;
211                 uint_t type;

213                 if (gelf_getsym(symdata, i, &sym) == NULL) {
214                         warnx("failed to get data about symbol %d", i);
215                         return (EXIT_FAILURE);
216                 }

218                 if ((name = elf_strptr(elf, symhdr.sh_link, sym.st_name)) ==
219                     NULL) {
220                         warnx("failed to get name for symbol %d", i);
221                         return (EXIT_FAILURE);
222                 }

224                 type = GELF_ST_TYPE(sym.st_info);
225                 if (type == STT_FILE) {
226                         curfile = name;
227                         continue;
228                 }

230                 if (strcmp(name, "global") == 0) {
231                         ret |= check_global(fp, &sym, i, curfile, name);
232                 } else if (strcmp(name, "mumble") == 0) {
233                         ret |= check_mumble(fp, &sym, i, curfile, name);
234                 }
235         }

237         return (ret);
238 }

240 int
241 main(int argc, char *argv[])
242 {
243         int i, ret = 0;

245         if (argc < 2) {
246                 errx(EXIT_FAILURE, "missing test files");
247         }

249         if (elf_version(EV_CURRENT) == EV_NONE) {
250                 errx(EXIT_FAILURE, "failed to initialize libelf");
251         }

253         for (i = 1; i < argc; i++) {
254                 int fd;
255                 ctf_file_t *fp;
256                 Elf *elf;

258                 if ((fd = open(argv[i], O_RDONLY)) < 0) {
```

```
259                         warn("failed to open %s", argv[i]);
260                         ret = EXIT_FAILURE;
261                         continue;
262                 }

264                 if ((elf = elf_begin(fd, ELF_C_READ, NULL)) == NULL) {
265                         warnx("failed to open libelf handle to %s", argv[i]);
266                         ret = EXIT_FAILURE;
267                         (void) close(fd);
268                         continue;
269                 }

271                 if ((fp = ctf_open(argv[i], &ret)) == NULL) {
272                         warnx("failed to open %s: %s", argv[i],
273                             ctf_errmsg(ret));
274                         ret = EXIT_FAILURE;
275                         (void) close(fd);
276                         (void) elf_end(elf);
277                         continue;
278                 }

280                 if (check_merge_static(argv[i], fp, elf) != 0) {
281                         ret = EXIT_FAILURE;
282                 }

284                 ctf_close(fp);
285                 (void) close(fd);
286                 (void) elf_end(elf);
287         }

289         return (ret);
290 }
```

```
**********************************************************
    1461 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/check-merge-weak.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Check that we properly handle weak symbols.
 */

#include "check-common.h"


static check_function_test_t functions[] = {
        { "mumble", "int", 0, 0, NULL },
        { "_mumble", "int", 0, 0, NULL },
        { NULL }
};

static check_symbol_t check_syms[] = {
        { "foo", "int" },
        { "_foo", "int" },
        { 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;
                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; functions[j].cft_name != NULL; j++) {
                        if (!ctftest_check_function(functions[j].cft_name, fp,
                            functions[j].cft_rtype, functions[j].cft_nargs,
```



```
                            functions[j].cft_flags, functions[j].cft_args)) {
                                ret = EXIT_FAILURE;
                        }
                }

                ctf_close(fp);
        }

        return (ret);
}
```

```
**********************************************************
    4692 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/check-reference.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

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

#include "check-common.h"

static check_number_t check_base[] = {
        { "char", CTF_K_INTEGER, CTF_INT_SIGNED | CTF_INT_CHAR, 0, 8 },
        { "int", CTF_K_INTEGER, CTF_INT_SIGNED, 0, 32 },
        { "float", CTF_K_FLOAT, CTF_FP_SINGLE, 0, 32 },
        { NULL }
};

static check_symbol_t check_syms[] = {
        { "a", "int" },
        { "aa", "test_int_t" },
        { "b", "const short" },
        { "c", "volatile float" },
        { "d", "int *" },
        { "dd", "int **" },
        { "ddd", "int ***" },
        { "e", "test_int_t *" },
        { "ce", "const test_int_t *" },
        { "ve", "volatile test_int_t *" },
        { "cve", "const volatile test_int_t *" },
        { "f", "int *const *" },
        { "g", "const char *const" },
        { NULL },
};

static check_descent_t check_descent_aa[] = {
        { "test_int_t", CTF_K_TYPEDEF },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_b[] = {
        { "const short", CTF_K_CONST },
        { "short", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_c[] = {
        { "volatile float", CTF_K_VOLATILE },
```



```
        { "float", CTF_K_FLOAT },
        { NULL }
};

static check_descent_t check_descent_d[] = {
        { "int *", CTF_K_POINTER },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_dd[] = {
        { "int **", CTF_K_POINTER },
        { "int *", CTF_K_POINTER },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_ddd[] = {
        { "int ***", CTF_K_POINTER },
        { "int **", CTF_K_POINTER },
        { "int *", CTF_K_POINTER },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_e[] = {
        { "test_int_t *", CTF_K_POINTER },
        { "test_int_t", CTF_K_TYPEDEF },
        { "int", CTF_K_INTEGER },
        { NULL },
};

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

static check_descent_t check_descent_ve[] = {
        { "volatile test_int_t *", CTF_K_POINTER},
        { "volatile test_int_t", CTF_K_VOLATILE },
        { "test_int_t", CTF_K_TYPEDEF },
        { "int", CTF_K_INTEGER },
        { NULL }
};

static check_descent_t check_descent_cve[] = {
        { "const volatile test_int_t *", CTF_K_POINTER },
        { "const volatile test_int_t", CTF_K_CONST },
        { "volatile test_int_t", CTF_K_VOLATILE },
        { "test_int_t", CTF_K_TYPEDEF },
        { "int", CTF_K_INTEGER },
        { NULL }
};

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

static check_descent_t check_descent_g[] = {
```

```
127         { "const char *const", CTF_K_CONST },
128         { "const char *", CTF_K_POINTER },
129         { "const char", CTF_K_CONST },
130         { "char", CTF_K_INTEGER },
131         { NULL }
132 };

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 };

146 static check_descent_test_t descents[] = {
147         { "aa", check_descent_aa },
148         { "b", check_descent_b },
149         { "c", check_descent_c },
150         { "d", check_descent_d },
151         { "dd", check_descent_dd },
152         { "ddd", check_descent_ddd },
153         { "e", check_descent_e },
154         { "ce", check_descent_ce },
155         { "ve", check_descent_ve },
156         { "cve", check_descent_cve },
157         { "f", check_descent_f },
158         { "g", check_descent_g },
159         { "cvh", check_descent_cvh },
160         { NULL }
161 };

163 int
164 main(int argc, char *argv[])
165 {
166         int i, ret = 0;

168         if (argc < 2) {
169                 errx(EXIT_FAILURE, "missing test files");
170         }

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

176                 if ((fp = ctf_open(argv[i], &ret)) == NULL) {
177                         warnx("failed to open %s: %s", argv[i],
178                             ctf_errmsg(ret));
179                         ret = EXIT_FAILURE;
180                         continue;
181                 }

183                 if (!ctftest_check_numbers(fp, check_base))
184                         ret = EXIT_FAILURE;
185                 if (!ctftest_check_symbols(fp, check_syms))
186                         ret = EXIT_FAILURE;
187                 for (d = 0; descents[d].cdt_sym != NULL; d++) {
188                         if (!ctftest_check_descent(descents[d].cdt_sym, fp,
189                             descents[d].cdt_tests)) {
190                                 ret = EXIT_FAILURE;
191                         }
192                 }
```



```
193                 ctf_close(fp);
194         }

196         return (ret);
197 }
```

```
**********************************************************
   11429 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/check-sou.c
10814 Want primordial CTF test suite
Reviewed 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) 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 }
};

static check_symbol_t check_syms[] = {
        { "foo", "struct foo" },
        { "head", "nlist_t" },
        { "forward", "const forward_t" },
        { "oot", "struct round_up" },
        { "botw", "struct fixed_up" },
        { "sophie", "struct mysterious_barrel" },
        { "ayesha", "struct dusk_barrel" },
        { "stats","struct stats" },
        { "ring", "struct fellowship" },
        { "rings", "struct rings" },
        { "nvme", "struct csts" },
```



```
        { "games", "union jrpg" },
        { "nier", "union nier" },
        { "kh", "union kh" },
        { "ct", "struct trigger" },
        { "regress", "const union regress [9]" },
        { NULL }
};

static check_member_t check_member_foo[] = {
        { "a", "int", 0 },
        { "b", "float", 4 * NBBY },
        { "c", "const char *", 8 * NBBY },
        { NULL }
};

static check_member_t check_member_node[] = {
        { "prev", "struct node *", 0 },
#ifdef  TARGET_LP64
        { "next", "struct node *", 8 * NBBY },
#else
        { "next", "struct node *", 4 * NBBY },
#endif
        { NULL }
};

static check_member_t check_member_nlist[] = {
        { "size", "size_t", 0 },
#ifdef  TARGET_LP64
        { "off", "size_t", 8 * NBBY },
        { "head", "struct node", 16 * NBBY },
#else
        { "off", "size_t", 4 * NBBY },
        { "head", "struct node", 8 * NBBY },
#endif
        { NULL }
};

static check_member_t check_member_forward[] = {
        { "past", "void *", 0 },
#ifdef  TARGET_LP64
        { "present", "void *", 8 * NBBY },
        { "future", "void *", 16 * NBBY },
#else
        { "present", "void *", 4 * NBBY },
        { "future", "void *", 8 * NBBY },
#endif
        { NULL }
};

static check_member_t check_member_round_up[] = {
        { "triforce", "uint8_t", 0 },
        { "link", "uint32_t", 4 * NBBY },
        { "zelda", "uint8_t", 8 * NBBY },
        { "ganon", "uint8_t", 9 * NBBY },
        { NULL }
};

static check_member_t check_member_fixed_up[] = {
        { "triforce", "uint8_t", 0 },
        { "link", "uint32_t", 1 * NBBY },
        { "zelda", "uint8_t", 5 * NBBY },
        { "ganon", "uint8_t", 6 * NBBY },
        { NULL }
};

#ifdef  TARGET_LP64
```

```
127 static check_member_t check_member_component[] = {
128         { "m", "enum material", 0 },
129         { "grade", "uint64_t", 8 * NBBY },
130         { "count", "uint64_t", 16 * NBBY },
131         { "locations", "const char *[4]", 24 * NBBY },
132         { NULL }
133 };

135 static check_member_t check_member_mysterious[] = {
136         { "name", "const char *", 0 },
137         { "capacity", "size_t", 8 * NBBY },
138         { "optional", "struct component [0]", 16 * NBBY },
139         { NULL }
140 };

142 static check_member_t check_member_dusk[] = {
143         { "name", "const char *", 0 },
144         { "opacity", "size_t", 8 * NBBY },
145         { "optional", "struct component [0]", 16 * NBBY },
146         { NULL }
147 };


150 static check_member_t check_member_stats[] = {
151         { "hp", "unsigned long:16", 0 },
152         { "mp", "unsigned long:16", 16 },
153         { "str", "unsigned long:8", 32 },
154         { "dex", "unsigned long:4", 40 },
155         { "con", "unsigned long:1", 44 },
156         { "inte", "unsigned long:2", 45 },
157         { "wis", "unsigned long:1", 47 },
158         { "cha", "unsigned long:4", 48 },
159         { "sanity", "unsigned long:1", 52 },
160         { "attack", "unsigned long:2", 53 },
161         { "mattack", "unsigned long:1", 55 },
162         { "defense", "unsigned long:8", 56 },
163         { "mdefense", "unsigned long:32", 64 },
164         { "evasion", "unsigned long:8", 96 },
165         { "crit", "unsigned long:5", 104 },
166         { "luck", "unsigned long:19", 109 },
167         { NULL }
168 };
169 #else
170 static check_member_t check_member_component[] = {
171         { "m", "enum material", 0 },
172         { "grade", "uint64_t", 4 * NBBY },
173         { "count", "uint64_t", 12 * NBBY },
174         { "locations", "const char *[4]", 20 * NBBY },
175         { NULL }
176 };

178 static check_member_t check_member_mysterious[] = {
179         { "name", "const char *", 0 },
180         { "capacity", "size_t", 4 * NBBY },
181         { "optional", "struct component [0]", 8 * NBBY },
182         { NULL }
183 };

185 static check_member_t check_member_dusk[] = {
186         { "name", "const char *", 0 },
187         { "opacity", "size_t", 4 * NBBY },
188         { "optional", "struct component [0]", 8 * NBBY },
189         { NULL }
190 };
```



```
193 static check_member_t check_member_stats[] = {
194         { "hp", "unsigned long long:16", 0 },
195         { "mp", "unsigned long long:16", 16 },
196         { "str", "unsigned long long:8", 32 },
197         { "dex", "unsigned long long:4", 40 },
198         { "con", "unsigned long long:1", 44 },
199         { "inte", "unsigned long long:2", 45 },
200         { "wis", "unsigned long long:1", 47 },
201         { "cha", "unsigned long long:4", 48 },
202         { "sanity", "unsigned long long:1", 52 },
203         { "attack", "unsigned long long:2", 53 },
204         { "mattack", "unsigned long long:1", 55 },
205         { "defense", "unsigned long long:8", 56 },
206         { "mdefense", "unsigned long long:32", 64 },
207         { "evasion", "unsigned long long:8", 96 },
208         { "crit", "unsigned long long:5", 104 },
209         { "luck", "unsigned long long:19", 109 },
210         { NULL }
211 };
212 #endif

214 static check_member_t check_member_fellowship[] = {
215         { "frodo", "unsigned short:1", 0 },
216         { "sam", "unsigned short:1", 1 },
217         { "merry", "unsigned short:1", 2 },
218         { "pippin", "unsigned short:1", 3 },
219         { "aragorn", "unsigned short:1", 4 },
220         { "boromir", "unsigned short:1", 5 },
221         { "legolas", "unsigned short:1", 6 },
222         { "gimli", "unsigned short:1", 7 },
223         { "gandalf", "unsigned short:1", 8 },
224         { NULL }
225 };

227 static check_member_t check_member_rings[] = {
228         { "elves", "unsigned int:3", 0 },
229         { "dwarves", "unsigned int:7", 3 },
230         { "men", "unsigned int:9", 10 },
231         { "one", "uint8_t", 3 * NBBY },
232         { "silmarils", "uint8_t [3]", 4 * NBBY },
233         { NULL }
234 };

236 static check_member_t check_member_csts[] = {
237         { "rdy", "unsigned int:7", 0 },
238         { "csts", "unsigned int:32", 7 },
239         { NULL }
240 };

242 static check_member_t check_member_jrpg[] = {
243         { "ff", "int", 0 },
244         { "atelier", "double [4]", 0 },
245         { "tales", "const char *", 0 },
246         { "chrono", "int (*)()", 0 },
247         { "xeno", "struct rings", 0 },
248         { NULL }
249 };

251 static check_member_t check_member_android[] = {
252         { "_2b", "unsigned int:16", 0 },
253         { "_9s", "unsigned int:16", 16 },
254         { NULL }
255 };

257 static check_member_t check_member_nier[] = {
258         { "automata", "uint32_t", 0 },
```

```
259         { "android", "struct android", 0 },
260         { NULL }
261 };

263 static check_member_t check_member_kh[] = {
264         { "sora", "int:3", 0 },
265         { "riku", "char:7", 0 },
266         { "kairi", "double", 0 },
267         { "namine", "complex double", 0 },
268         { NULL }
269 };

271 static check_member_t check_member_trigger[] = {
272         { "chrono", "uint8_t", 0 },
273         { "cross", "uint8_t", 8 },
274         /*
275          * This test has an anonymous union. Unfortunately, there's not a great
276          * way to distinguish between various anonymous unions in this form.
277          */
278 #ifdef  TARGET_LP64
279         { "", "union ", 64 },
280 #else
281         { "", "union ", 32 },
282 #endif
283         { NULL }
284 };

286 static check_member_t check_member_regress[] = {
287         { "i", "unsigned int [3]", 0 },
288         { "e", "long double", 0 },
289         { NULL }
290 };

292 static check_member_test_t members[] = {
293 #ifdef  TARGET_LP64
294         { "struct foo", CTF_K_STRUCT, 16, check_member_foo },
295         { "struct node", CTF_K_STRUCT, 16, check_member_node },
296         { "struct nlist", CTF_K_STRUCT, 32, check_member_nlist },
297         { "struct forward", CTF_K_STRUCT, 24, check_member_forward },
298 #else
299         { "struct foo", CTF_K_STRUCT, 12, check_member_foo },
300         { "struct node", CTF_K_STRUCT, 8, check_member_node },
301         { "struct nlist", CTF_K_STRUCT, 16, check_member_nlist },
302         { "struct forward", CTF_K_STRUCT, 12, check_member_forward },
303 #endif
304         { "struct round_up", CTF_K_STRUCT, 12, check_member_round_up },
305         { "struct fixed_up", CTF_K_STRUCT, 7, check_member_fixed_up },
306 #ifdef  TARGET_LP64
307         { "struct component", CTF_K_STRUCT, 56, check_member_component },
308         { "struct mysterious_barrel", CTF_K_STRUCT, 16,
309             check_member_mysterious },
310         { "struct dusk_barrel", CTF_K_STRUCT, 16, check_member_dusk },
311 #else
312         { "struct component", CTF_K_STRUCT, 36, check_member_component },
313         { "struct mysterious_barrel", CTF_K_STRUCT, 8,
314           check_member_mysterious },
315         { "struct dusk_barrel", CTF_K_STRUCT, 8, check_member_dusk },
316 #endif
317         { "struct stats", CTF_K_STRUCT, 16, check_member_stats },
318         { "struct fellowship", CTF_K_STRUCT, 2, check_member_fellowship },
319         { "struct rings", CTF_K_STRUCT, 8, check_member_rings },
320         { "struct csts", CTF_K_STRUCT, 5, check_member_csts },
321         { "union jrpg", CTF_K_UNION, 32, check_member_jrpg },
322         { "struct android", CTF_K_STRUCT, 4, check_member_android },
323         { "union nier", CTF_K_UNION, 4, check_member_nier },
324         { "union kh", CTF_K_UNION, 16, check_member_kh },
```



```
325 #ifdef  TARGET_LP64
326         { "struct trigger", CTF_K_STRUCT, 32, check_member_trigger },
327         { "union regress", CTF_K_UNION, 16, check_member_regress },
328 #else
329         { "struct trigger", CTF_K_STRUCT, 28, check_member_trigger },
330         { "union regress", CTF_K_UNION, 12, check_member_regress },
331 #endif
332         { NULL }
333 };

335 static check_descent_t check_descent_head[] = {
336         { "nlist_t", CTF_K_TYPEDEF },
337         { "struct nlist", CTF_K_STRUCT },
338         { NULL }
339 };

341 static check_descent_t check_descent_forward[] = {
342         { "const forward_t", CTF_K_CONST },
343         { "forward_t", CTF_K_TYPEDEF },
344         { "struct forward", CTF_K_STRUCT },
345         { NULL }
346 };

348 static check_descent_t check_descent_regress[] = {
349         { "const union regress [9]", CTF_K_CONST },
350         { "union regress [9]", CTF_K_ARRAY, "union regress", 9 },
351         { "union regress", CTF_K_UNION },
352         { NULL }
353 };

355 static check_descent_test_t descents[] = {
356         { "head", check_descent_head },
357         { "forward", check_descent_forward },
358         { "regress", check_descent_regress },
359         { NULL }
360 };

362 int
363 main(int argc, char *argv[])
364 {
365         int i, ret = 0;

367         if (argc < 2) {
368                 errx(EXIT_FAILURE, "missing test files");
369         }

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

375                 if ((fp = ctf_open(argv[i], &ret)) == NULL) {
376                         warnx("failed to open %s: %s", argv[i],
377                             ctf_errmsg(ret));
378                         ret = EXIT_FAILURE;
379                         continue;
380                 }

382                 if (!ctftest_check_numbers(fp, check_bitfields))
383                         ret = EXIT_FAILURE;
384                 if (!ctftest_check_symbols(fp, check_syms))
385                         ret = EXIT_FAILURE;
386                 for (j = 0; descents[j].cdt_sym != NULL; j++) {
387                         if (!ctftest_check_descent(descents[j].cdt_sym, fp,
388                             descents[j].cdt_tests)) {
389                                 ret = EXIT_FAILURE;
390                         }
```

```
391                 }

393                 for (j = 0; members[j].cmt_type != NULL; j++) {
394                         if (!ctftest_check_members(members[j].cmt_type, fp,
395                             members[j].cmt_kind, members[j].cmt_size,
396                             members[j].cmt_members)) {
397                                 ret = EXIT_FAILURE;
398                         }
399                 }

401                 ctf_close(fp);
402         }

404         return (ret);
405 }
```

```
**********************************************************
    1467 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/check-weak.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Check that we properly handle weak symbols.
 */

#include "check-common.h"


static check_function_test_t functions[] = {
        { "mumble", "int", 0, 0, NULL },
        { "_mumble", "int", 0, 0, NULL },
        { NULL }
};

static check_symbol_t check_syms[] = {
        { "strong", "int" },
        { "_strong", "int" },
        { 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;
                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; functions[j].cft_name != NULL; j++) {
                        if (!ctftest_check_function(functions[j].cft_name, fp,
                            functions[j].cft_rtype, functions[j].cft_nargs,
```



```
                            functions[j].cft_flags, functions[j].cft_args)) {
                                ret = EXIT_FAILURE;
                        }
                }

                ctf_close(fp);
        }

        return (ret);
}
```

```
**********************************************************
    5496 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/ctftest.ksh
10814 Want primordial CTF test suite
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
**********************************************************
```

```
#!/usr/bin/ksh
#
#
# 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) 2019, Joyent, Inc.
#

#
# Run all of the various CTF tests
#

unalias -a
#set -o xtrace

if [[ -z "$TMPDIR" ]]; then
        TMPDIR="/tmp"
fi

ctf_arg0=$(basename $0)
ctf_root=$(cd $(dirname $0) && echo $PWD)
ctf_tests=
ctf_compiler="gcc"
ctf_convert="ctfconvert"
ctf_merge="ctfmerge"
ctf_debugflags="-gdwarf-2 "
ctf_mach32="-m32"
ctf_mach64="-m64"
ctf_32cflags="$ctf_mach32 $ctf_debugflags"
ctf_64cflags="$ctf_mach64 $ctf_debugflags"
ctf_temp="$TMPDIR/ctftest.$$.o"
ctf_makefile="Makefile.ctftest"
ctf_nerrs=0

usage()
{
        typeset msg="$*"
        [[ -z "$msg" ]] || echo "$msg" >&2
        cat <<USAGE >&2
Usage: $ctf_arg0  [-c compiler] [-g flags] [-m ctfmerge] [-t ctfconvert]

        Runs the CTF test suite

        -c compiler             Use the specified compiler, defaults to 'gcc'
                                on path.
        -g flags                Use flags to generate debug info. Defaults to
                                "-gdwarf-2".
        -m ctfmerge             Use the specified ctfmerge, defaults to
                                'ctfmerge' on path.
        -t ctfconvert           Use the specified ctfconvert, defaults to
                                'ctfconvert' on path.
```



```
USAGE
        exit 2
}


test_fail()
{
        typeset msg="$*"
        [[ -z "$msg" ]] && msg="failed"
        echo "TEST FAILED: $msg" >&2
        ((ctf_nerrs++))
}

fatal()
{
        typeset msg="$*"
        [[ -z "$msg" ]] && msg="failed"
        echo "$ctf_arg0: $msg" >&2
        rm -f "$ctf_tmp32" "$ctf_temp64"
        exit 1
}

check_env()
{
        if which "$1" 2>/dev/null >/dev/null; then
                return
        fi

        [[ -f "$1" ]] || fatal "failed to find tool $1"
}

announce()
{
        cat << EOF
Beginning CTF tests with the following settings:
COMPILER:       $(which $ctf_compiler)
CTFCONVERT:     $(which $ctf_convert)
CTFMERGE:       $(which $ctf_merge)
32-bit CFLAGS:  $ctf_32cflags
64-bit CFLAGS:  $ctf_64cflags

EOF
}

run_one()
{
        typeset source=$1 checker=$2 flags=$3

        if ! "$ctf_compiler" $flags -o "$ctf_temp" -c "$source"; then
                test_fail "failed to compile $source with flags: $flags"
                return
        fi

        if ! "$ctf_convert" "$ctf_temp"; then
                test_fail "failed to convert CTF in $source"
                return
        fi

        if ! "$checker" "$ctf_temp"; then
                test_fail "check for $source, $checker, failed"
                return
        fi

        echo "TEST PASSED: $source $flags"
}
```

```
#
# Perform a more complex build. The Makefile present will drive the
# building of the artifacts and the running of the tests based on the
# variables that we pass to it.
#
run_dir()
{
        typeset dir outdir check32 check64 flags32 flags64

        dir=$1
        outdir="$TMPDIR/ctftest.$$-$(basename $d)"
        check32=$2
        flags32=$3
        check64=$4
        flags64=$5

        if ! mkdir $outdir; then
                fatal "failed to make temporary directory '$outdir'"
        fi

        if ! make -C $dir -f Makefile.ctftest \
            BUILDDIR="$outdir" \
            CC="$ctf_compiler" \
            CFLAGS32="$ctf_mach32" \
            CFLAGS64="$ctf_mach64" \
            DEBUGFLAGS="$ctf_debugflags" \
            CTFCONVERT="$ctf_convert" \
            CTFMERGE="$ctf_merge" \
            build 1>/dev/null; then
                rm -rf $outdir
                test_fail "failed to build $dir"
                return
        fi

        if ! make -C $dir -f Makefile.ctftest \
            BUILDDIR="$outdir" \
            CHECK32="$check32" \
            CHECK64="$check64" \
            run-test 1>/dev/null; then
                rm -rf $outdir
                test_fail "failed to run tests for $dir"
                return
        fi

        rm -rf $outdir
        echo "TEST PASSED: $dir (dir)"
}

#
# Find all of the tests that exist and then try to run them all. Tests
# may either be a single file or a directory.
#
run_tests()
{
        typeset t base check
        ctf_tests=$(ls "$ctf_root"/*.c)
        for t in $ctf_tests; do
                base=$(basename "$t" .c)
                check=$(echo "$base" | sed s/test-/check-/)
                if [[ -f "$ctf_root/$check" ]]; then
                        run_one $t "$ctf_root/$check" "$ctf_32cflags"
                        run_one $t "$ctf_root/$check" "$ctf_64cflags"
                elif [[ -f "$ctf_root/$check-32" && \
                    -f "$ctf_root/$check-64" ]]; then
                        run_one $t "$ctf_root/$check-32" "$ctf_32cflags"
                        run_one $t "$ctf_root/$check-64" "$ctf_64cflags"
```



```
                else
                        test_fail "missing checker for $t"
                fi
        done

        for d in $(find "$ctf_root" -maxdepth 1 -type d -name 'test-*'); do
                [[ ! -f "$d/$ctf_makefile" ]] && continue
                base=$(basename "$d")
                check=$(echo "$base" | sed s/test-/check-/)
                if [[ -f "$ctf_root/$check" ]]; then
                        run_dir $d "$ctf_root/$check" "$ctf_32cflags" \
                            "$ctf_root/$check" "$ctf_64cflags"
                elif [[ -f "$ctf_root/$check-32" && \
                    -f "$ctf_root/$check-64" ]]; then
                        run_dir $d "$ctf_root/$check-32" "$ctf_32cflags" \
                            "$ctf_root/$check-64" "$ctf_64cflags"
                else
                        test_fail "missing checker for $t"
                fi
        done
}

while getopts ":c:g:m:t:" c $@; do
        case "$c" in
        c)
                ctf_compiler=$OPTARG
                ;;
        g)
                ctf_debugflags=$OPTARG
                ;;
        m)
                ctf_merge=$OPTARG
                ;;
        t)
                ctf_convert=$OPTARG
                ;;
        :)
                usage "option requires an argument -- $OPTARG"
                ;;
        *)
                usage "invalid option -- $OPTARG"
                ;;
        esac
done

ctf_32cflags="$ctf_mach32 $ctf_debugflags"
ctf_64cflags="$ctf_mach64 $ctf_debugflags"

check_env "$ctf_compiler"
check_env "$ctf_convert"
check_env "$ctf_merge"
announce

run_tests

if [[ $ctf_nerrs -ne 0 ]]; then
        if [[ $ctf_nerrs -eq 1 ]]; then
                printf "\n%s: %u test failed\n" "$ctf_arg0" "$ctf_nerrs"
        else
                printf "\n%s: %u tests failed\n" "$ctf_arg0" "$ctf_nerrs"
        fi
else
        printf "\n%s: All tests passed successfully\n" "$ctf_arg0"
        exit 0
fi
```

```
**********************************************************
     712 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/test-array.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * A series of basic array tests with simple base types.
 */

int a[3];
double b[42];
const char *c[] = { "17" "31", "169" };

int d[4][5];
int e[4][5][6];
int f[4][5][6][7];
int g[4][5][6][7][8];
int h[4][5][6][7][8][9];
int i[4][5][6][7][8][9][10];
```

```
**********************************************************
    1464 Tue Apr 23 05:35:50 2019
new/usr/src/test/util-tests/tests/ctf/test-enum.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <stdint.h>

/*
 * Basic sanity checking of enumerations, using specific numbers and arbitrary
 * numbers.
 */

enum ff6 {
        TERRA,
        LOCKE,
        EDGAR,
        SABIN,
        CELES,
        CYAN,
        SHADOW,
        GAU,
        SETZER,
        STRAGO,
        RELM,
        MOG,
        GOGO,
        UMARO,
        LEO,
        KEFKA
};

typedef enum ff10 {
        TIDUS = -10,
        YUNA = 23,
        AURON = -34,
        WAKA = 52,
        LULU = INT32_MAX,
        RIKKU = INT32_MIN,
        KHIMARI = 0
} ff10_t;

/*
 * The following enum is copy of the ddi_hp_cn_state_t enumeration which was
 * previously incorrectly converted by the tools. Notably, we always assumed
 * that the DWARF enum values were signed; however, in this case we needed to
 * check for an unsigned version before a signed version, otherwise some of the
 * entries below will have the wrong values.
 */
typedef enum chrono {
        CRONO = 0x1000,
```



```
        LUCCA = 0x2000,
        MARLE = 0x3000,
        FROG = 0x4000,
        ROBO = 0x5000,
        AYLA = 0x6000,
        MAGUS = 0x7000,
        SCHALA = 0x8000,
        LAVOS = 0x9000,
        BALTHAZAR = 0xa000
} chrono_t;

enum ff6 ff6;
ff10_t ff10;
chrono_t trigger;
```

```
**********************************************************
     715 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-float.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <complex.h>

/*
 * Test floating point types. Unfortunately neither gcc nor clang support the
 * imaginary keyword which means that we cannot test it.
 */

float a;
double b;
long double c;
float complex d;
double complex e;
long double complex f;
```

```
**********************************************************
     732 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-forward.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * This tests the use of forward declarations of unknown types.
 */

struct foo;
union bar;
enum baz;

struct forward {
        struct foo *prev;
        struct foo *next;
        union bar *data;
        enum baz *tag;
};

struct foo *foop;
union bar *barp;
enum baz *bazp;
struct forward forward;
```

```
**********************************************************
    1143 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-function.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <sys/types.h>
#include <string.h>

/*
 * Test various function and function pointer cases.
 */

static void
simple_func(void)
{
}

static void
one(int v)
{
}

static void
two(int v, const char *a)
{
}

static void
three(int v, const char *a, float b)
{
}

static const char *
noarg(void)
{
        return ("hello, world");
}

static const char *
argument(uintptr_t base)
{
        return ((const char *)(base + 1));
}

static void
vararg(const char *foo, ...)
{

}
```



```
static uintptr_t
vararg_ret(const char *foo, ...)
{
        return ((uintptr_t)foo);
}

typedef int (*strfunc_t)(const char *, const char *);
typedef void (*vararg_t)(const char *, ...);

strfunc_t s = strcmp;
vararg_t v = vararg;
```

```
**********************************************************
     702 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-int.c
10814 Want primordial CTF test suite
Reviewed 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 (c) 2019, Joyent, Inc.
  14  */

  16 /*
  17  * Test basic integer types. Note that signed types are considered the same as
  18  * the base type.
  19  */

  21 char a;
  22 unsigned char b;

  24 short d;
  25 unsigned short e;

  27 int g;
  28 unsigned int h;

  30 long j;
  31 unsigned long k;

  33 long long m;
  34 unsigned long long n;
```

```
**********************************************************
    1126 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-dedup/Makefile.ctftest
10814 Want primordial CTF test suite
Reviewed 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.
#

TEST =          test-merge-dedup

OBJS_C_32 =     $(BUILDDIR)/test-merge-1.32.c.o \
                $(BUILDDIR)/test-merge-2.32.c.o \
                $(BUILDDIR)/test-merge-3.32.c.o \
                $(BUILDDIR)/test-merge-dedup.32.c.o

OBJS_C_64 =     $(BUILDDIR)/test-merge-1.64.c.o \
                $(BUILDDIR)/test-merge-2.64.c.o \
                $(BUILDDIR)/test-merge-3.64.c.o \
                $(BUILDDIR)/test-merge-dedup.64.c.o

OBJS_M_32 =     $(BUILDDIR)/test-merge-1.32.m.o \
                $(BUILDDIR)/test-merge-2.32.m.o \
                $(BUILDDIR)/test-merge-3.32.m.o \
                $(BUILDDIR)/test-merge-dedup.32.m.o

OBJS_M_64 =     $(BUILDDIR)/test-merge-1.64.m.o \
                $(BUILDDIR)/test-merge-2.64.m.o \
                $(BUILDDIR)/test-merge-3.64.m.o \
                $(BUILDDIR)/test-merge-dedup.64.m.o


include         ../Makefile.ctftest.com
```

```
**********************************************************
     564 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-dedup/test-merge-1.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

int f;
short g;
const char *h;
float i;
double j;

struct dup {
        int dup;
        int dup2;
};

struct dup dup1;
```

```
**********************************************************
     564 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-dedup/test-merge-2.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

int k;
short l;
const char *m;
float n;
double o;

struct dup {
        int dup;
        int dup2;
};

struct dup dup2;
```

```
**********************************************************
     564 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-dedup/test-merge-3.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

int p;
short q;
const char *r;
float s;
double t;

struct dup {
        int dup;
        int dup2;
};

struct dup dup3;
```

```
**********************************************************
     624 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-dedup/test-merge-dedup.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

int a;
short b;
const char *c;
float d;
double e;

struct dup {
        int dup;
        int dup2;
};

struct dup dupmain;

int
main(int argc, const char *argv[])
{
        return (0);
}
```

```
**********************************************************
     804 Tue Apr 23 05:35:51 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-forward/Makefile.ctftest
10814 Want primordial CTF test suite
Reviewed 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.c.org/license/CDDL.
#

#
# Copyright 2019 Joyent, Inc.
#

TEST =          test-merge-forward

OBJS_C_32 =     $(BUILDDIR)/test-merge.32.c.o \
                $(BUILDDIR)/test-impl.32.c.o

OBJS_C_64 =     $(BUILDDIR)/test-merge.64.c.o \
                $(BUILDDIR)/test-impl.64.c.o

OBJS_M_32 =     $(BUILDDIR)/test-merge.32.m.o \
                $(BUILDDIR)/test-impl.32.m.o

OBJS_M_64 =     $(BUILDDIR)/test-merge.64.m.o \
                $(BUILDDIR)/test-impl.64.m.o

include ../Makefile.ctftest.com
```

****
642 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-forward/test-impl.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

struct foo {
        struct foo *next;
        int left;
        int right;
        int count;
};

void
mumble(struct foo *foo)
{
        foo->left = foo->right - foo->count;
        foo->count += foo->right;
        foo->right--;
}
```

```
**********************************************************
     648 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-forward/test-merge.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <stdio.h>

struct foo;

typedef struct foo_list {
        int count;
        struct foo *head;
        struct foo *tail;
} foo_list_t;

foo_list_t list;

int
main(void)
{
        (void) printf("%p", &list);
}
```

```
**********************************************************
    2829 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-reduction/Makefile.ctftest
10814 Want primordial CTF test suite
Reviewed 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.
#

#
# This makefile could be simplified substantially. However, it does
# everything explicitly to try and work with a wide variety of different
# makes.
#
# The following values should be passed in by the invoker of the
# Makefile:
#
#       CC              C Compiler to use
#       CFLAGS32        32-bit CFLAGS
#       CFLAGS64        64-bit CFLAGS
#       CTFCONVERT      Path to ctfconvert
#       CTFMERGE        Path to ctfmerge
#       DEBUGFLAGS      The set of debug flags to use
#       BUILDDIR        Directory things should be built in
#       CHECK32         Program to check 32-bit output
#       CHECK64         Program to check 64-bit output
#

OBJS_C_32 =     $(BUILDDIR)/test-global.32.c.o \
                $(BUILDDIR)/test-scoped.32.c.o

OBJS_C_64 =     $(BUILDDIR)/test-global.64.c.o \
                $(BUILDDIR)/test-scoped.64.c.o

OBJS_M_32 =     $(BUILDDIR)/test-global.32.m.o \
                $(BUILDDIR)/test-scoped.32.m.o

OBJS_M_64 =     $(BUILDDIR)/test-global.64.m.o \
                $(BUILDDIR)/test-scoped.64.m.o

BINS =          $(BUILDDIR)/test-merge-reduction-32c.so.1 \
                $(BUILDDIR)/test-merge-reduction-32m.so.1 \
                $(BUILDDIR)/test-merge-reduction-64c.so.1 \
                $(BUILDDIR)/test-merge-reduction-64m.so.1 \

CFLAGS = -fPIC
LDFLAGS = -shared -Wl,-Mmapfile-vers -Wl,-ztext -Wl,-zdefs \
          -htest-merge-reduction.so.1

build: $(BINS)

$(BUILDDIR)/%.32.c.o: %.c
        $(CC) $(CFLAGS) $(CFLAGS32) $(DEBUGFLAGS) -o $@ -c $<
```



```
$(BUILDDIR)/%.64.c.o: %.c
        $(CC) $(CFLAGS) $(CFLAGS64) $(DEBUGFLAGS) -o $@ -c $<

$(BUILDDIR)/%.32.m.o: %.c
        $(CC) $(CFLAGS) $(CFLAGS32) $(DEBUGFLAGS) -o $@ -c $<
        $(CTFCONVERT) $@

$(BUILDDIR)/%.64.m.o: %.c
        $(CC) $(CFLAGS) $(CFLAGS64) $(DEBUGFLAGS) -o $@ -c $<
        $(CTFCONVERT) $@

$(BUILDDIR)/test-merge-reduction-32c.so.1: $(OBJS_C_32)
        $(CC) $(CFLAGS32) $(CFLAGS) $(LDFLAGS) $(DEBUGFLAGS) -o $@ $(OBJS_C_32)
        $(CTFCONVERT) $@

$(BUILDDIR)/test-merge-reduction-64c.so.1: $(OBJS_C_64)
        $(CC) $(CFLAGS64) $(CFLAGS) $(LDFLAGS) $(DEBUGFLAGS) -o $@ $(OBJS_C_64)
        $(CTFCONVERT) $@

$(BUILDDIR)/test-merge-reduction-32m.so.1: $(OBJS_M_32)
        $(CC) $(CFLAGS32) $(CFLAGS) $(LDFLAGS) $(DEBUGFLAGS) -o $@ $(OBJS_M_32)
        $(CTFMERGE) -t -o $@ $(OBJS_M_32)

$(BUILDDIR)/test-merge-reduction-64m.so.1: $(OBJS_M_64)
        $(CC) $(CFLAGS64) $(CFLAGS) $(LDFLAGS) $(DEBUGFLAGS) -o $@ $(OBJS_M_64)
        $(CTFMERGE) -t -o $@ $(OBJS_M_64)

run-test:
        $(CHECK32) $(BUILDDIR)/test-merge-reduction-32c.so.1
        $(CHECK64) $(BUILDDIR)/test-merge-reduction-64c.so.1
        $(CHECK32) $(BUILDDIR)/test-merge-reduction-32m.so.1
        $(CHECK64) $(BUILDDIR)/test-merge-reduction-64m.so.1
```

```
**********************************************************
     871 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-reduction/mapfile-vers
10814 Want primordial CTF test suite
Reviewed 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.
#

#
# MAPFILE HEADER START
#
# WARNING:  STOP NOW.  DO NOT MODIFY THIS FILE.
# Object versioning must comply with the rules detailed in
#
#       usr/src/lib/README.mapfiles
#
# You should not be making modifications here until you've read the most current
# copy of that file. If you need help, contact a gatekeeper for guidance.
#
# MAPFILE HEADER END
#

$mapfile_version 2

SYMBOL_VERSION CTFTEST {
    global:
        global;
    local:
        *;
};
```

****
563 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-reduction/test-global.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */


#include <stdlib.h>

extern int scoped(uint32_t);

int
global(void)
{
        return (scoped(arc4random()));
}
```

****
607 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-reduction/test-scoped.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <sys/types.h>
#include <limits.h>

int data;

int
scoped(uint32_t a)
{
        if (a >= INT32_MAX) {
                data = a - INT32_MAX;
        }

        return (data);
}
```

```
**********************************************************
    1146 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-static/Makefile.ctftest
10814 Want primordial CTF test suite
Reviewed 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.
#

TEST =          test-merge-static

OBJS_C_32 =     $(BUILDDIR)/test-a.32.c.o \
                $(BUILDDIR)/test-b.32.c.o \
                $(BUILDDIR)/test-c.32.c.o \
                $(BUILDDIR)/test-d.32.c.o \
                $(BUILDDIR)/test-main.32.c.o

OBJS_C_64 =     $(BUILDDIR)/test-a.64.c.o \
                $(BUILDDIR)/test-b.64.c.o \
                $(BUILDDIR)/test-c.64.c.o \
                $(BUILDDIR)/test-d.64.c.o \
                $(BUILDDIR)/test-main.64.c.o

OBJS_M_32 =     $(BUILDDIR)/test-a.32.m.o \
                $(BUILDDIR)/test-b.32.m.o \
                $(BUILDDIR)/test-c.32.m.o \
                $(BUILDDIR)/test-d.32.m.o \
                $(BUILDDIR)/test-main.32.m.o

OBJS_M_64 =     $(BUILDDIR)/test-a.64.m.o \
                $(BUILDDIR)/test-b.64.m.o \
                $(BUILDDIR)/test-c.64.m.o \
                $(BUILDDIR)/test-d.64.m.o \
                $(BUILDDIR)/test-main.64.m.o

include         ../Makefile.ctftest.com
```

****
556 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-static/test-a.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <sys/types.h>

static uint8_t global;

static uint8_t
mumble(uint8_t a)
{
        return (a);
}
```

```
**********************************************************
     559 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-static/test-b.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <sys/types.h>

static uint16_t global;

static uint16_t
mumble(uint16_t a)
{
        return (a);
}
```

```
**********************************************************
     559 Tue Apr 23 05:35:52 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-static/test-c.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <sys/types.h>

static uint32_t global;

static uint32_t
mumble(uint32_t a)
{
        return (a);
}
```

```
**********************************************************
     559 Tue Apr 23 05:35:53 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-static/test-d.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <sys/types.h>

static uint64_t global;

static uint64_t
mumble(uint64_t a)
{
        return (a);
}
```

```
**********************************************************
     539 Tue Apr 23 05:35:53 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-static/test-main.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

int global;

int
mumble(int a)
{
        return (a);
}

int
main(void)
{
        return (0);
}
```

```
**********************************************************
     686 Tue Apr 23 05:35:53 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-weak/Makefile.ctftest
10814 Want primordial CTF test suite
Reviewed 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.c.org/license/CDDL.
#

#
# Copyright 2019 Joyent, Inc.
#

TEST =          test-merge-weak

OBJS_C_32 =     $(BUILDDIR)/test-merge-weak.32.c.o
OBJS_C_64 =     $(BUILDDIR)/test-merge-weak.64.c.o
OBJS_M_32 =     $(BUILDDIR)/test-merge-weak.32.m.o
OBJS_M_64 =     $(BUILDDIR)/test-merge-weak.64.m.o

include ../Makefile.ctftest.com
```

****
641 Tue Apr 23 05:35:53 2019
new/usr/src/test/util-tests/tests/ctf/test-merge-weak/test-merge-weak.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <stdlib.h>

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

int _foo = 5;

int
_mumble(void)
{
        return ((int)arc4random());
}

int
main(void)
{
        return (mumble());
};
```

```
**********************************************************
     961 Tue Apr 23 05:35:53 2019
new/usr/src/test/util-tests/tests/ctf/test-reference.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

/*
 * Test the encoding of references to another type. Specifically the references
 * that we generally care about are things like:
 *
 * o pointers
 * o typedefs
 * o const
 * o volatile
 * o restrict
 */

int a;
typedef int test_int_t;
test_int_t aa;
const short b;
volatile float c;

int *d;
int **dd;
int ***ddd;
test_int_t *e;
const test_int_t *ce;
volatile test_int_t *ve;
volatile const test_int_t *cve;
int *const *f;
const char *const g;

typedef int *const * foo_t;
const volatile foo_t *cvh;
```

```
**********************************************************
    3777 Tue Apr 23 05:35:53 2019
new/usr/src/test/util-tests/tests/ctf/test-sou.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#include <sys/types.h>
#include <complex.h>

/*
 * Test various structure and union constructs, including various things that
 * have caused regressions in the past.
 */

/*
 * Basic, simple struct.
 */
struct foo {
        int a;
        float b;
        const char *c;
};

struct foo foo;

/*
 * Self-referential structs
 */
struct node {
        struct node *prev;
        struct node *next;
};

typedef struct nlist {
        size_t size;
        size_t off;
        struct node head;
} nlist_t;

nlist_t head;

/*
 * Struct that has a forward declaration.
 */
typedef struct forward forward_t;
struct forward {
        void *past;
        void *present;
        void *future;
};
```



```
const forward_t forward;

/*
 * Here, we have a pair of structures that basically round up to different
 * sizes. As in, the size of the structure is somewhat compiler dependent.
 */
struct round_up {
        uint8_t triforce;
        uint32_t link;
        uint8_t zelda;
        uint8_t ganon;
};

#pragma pack(1)
struct fixed_up {
        uint8_t triforce;
        uint32_t link;
        uint8_t zelda;
        uint8_t ganon;
};
#pragma pack()

struct round_up oot;
struct fixed_up botw;

/*
 * Various GNU and c99 style arrays
 */
enum material {
        COPPER,
        IRON,
        STEEL,
        ADAMANTIUM,
        MYTHRIL,
        ORIHALCUM
};

struct component {
        enum material m;
        uint64_t grade;
        uint64_t count;
        const char *locations[4];
};

struct mysterious_barrel {
        const char *name;
        size_t capacity;
        struct component optional[];
};

struct dusk_barrel {
        const char *name;
        size_t opacity;
        struct component optional[0];
};

struct mysterious_barrel sophie;
struct dusk_barrel ayesha;

/*
 * Various bitfield forms.
 */

/*
 * Variant of the Intel system_desc.
 */
```

```
127 struct stats {
128         uint64_t hp:16;
129         uint64_t mp:16;
130         uint64_t str:8;
131         uint64_t dex:4;
132         uint64_t con:1;
133         uint64_t inte:2;
134         uint64_t wis:1;
135         uint64_t cha:4;
136         uint64_t sanity:1;
137         uint64_t attack:2;
138         uint64_t mattack:1;
139         uint64_t defense:8;
140         uint64_t mdefense:32;
141         uint64_t evasion:8;
142         uint64_t crit:5;
143         uint64_t luck:19;
144 };

146 struct stats stats;

148 /*
149  * More odd length structures due to bitfields
150  */
151 struct fellowship {
152         uint16_t frodo:1;
153         uint16_t sam:1;
154         uint16_t merry:1;
155         uint16_t pippin:1;
156         uint16_t aragorn:1;
157         uint16_t boromir:1;
158         uint16_t legolas:1;
159         uint16_t gimli:1;
160         uint16_t gandalf:1;
161 };

163 struct fellowship ring;

165 struct rings {
166         uint32_t elves:3;
167         uint32_t dwarves:7;
168         uint32_t men:9;
169         uint8_t one;
170         uint8_t silmarils[3];
171 };

173 struct rings rings;

175 /*
176  * Regression, we didn't handle receiving a negative offset from DWARF with
177  * this.
178  */
179 #pragma pack(1)
180 struct csts {
181         unsigned int rdy:7;
182         unsigned int csts:32;
183 };

185 struct csts nvme;
186 #pragma pack()

188 /*
189  * Onto unions
190  */
191 union jrpg {
192         int ff;
```



```
193         double atelier[4];
194         const char *tales;
195         int (*chrono)(void);
196         struct rings xeno;
197 };

199 union jrpg games;

201 #pragma pack(1)
202 struct android {
203         uint32_t _2b:16;
204         uint32_t _9s:16;
205 };

207 union nier {
208         uint32_t automata;
209         struct android android;
210 };
211 #pragma pack()

213 union nier nier;

215 union kh {
216         int sora:3;
217         char riku:7;
218         double kairi;
219         complex double namine;
220 };

222 union kh kh;

224 /*
225  * Anonymous union in a struct, GNU extension / C11
226  */

228 struct trigger {
229         uint8_t chrono;
230         uint8_t cross;
231         union {
232                 void *lavos;
233                 int *crono;
234                 uint64_t schala[3];
235         };
236 };

238 struct trigger ct;

240 /*
241  * This is an array/union combo that failed conversion previously.
242  */
243 static const union regress {
244         unsigned int i[3];
245         long double e;
246 } regress[9];
```

```
**********************************************************
     572 Tue Apr 23 05:35:53 2019
new/usr/src/test/util-tests/tests/ctf/test-weak.c
10814 Want primordial CTF test suite
Reviewed 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) 2019, Joyent, Inc.
 */

#pragma weak  _strong = strong
#pragma weak _mumble = mumble

int strong = 3;

int
mumble(void)
{
        return (42);
}
```