Gnu Gcc Testsuite

PLCT Lab
Molly Chen
xiaoou@iscas.ac.cn

2021.3.9

Content

- **Running GCC testsuite**
  - **Testing a cross-compiler**
- Testsuite organization
- Adding new tests

Running GCC testsuite (1/5)

- Optinal, but give you confidence or point out problems before using your new GCC installation.
- Include in the full distribution. If not, download the testsuites:
  - https://github.com/gcc-mirror/gcc/tree/master/gcc/testsuite
- Prerequisites:
  - DejaGnu[1], Tcl, and Expect, Python3 and pytest module.
  - HSAILasm for BRIG frontend tests

[1] include in the full distribution of gcc toolchain

Running GCC testsuite (2/5)

- Run testsuite:

```
cd objdir; make check
```

- Run on selected tests
  - Select targets: `make check-gcc`
  - Select language: `make check-c, make check-c++, make check-d`

    `make check-fortran, make check-ada, make check-objc`
  - Run all gcc execute tests: `make check-gcc RUNTESTFLAGS="execute.exp`[1] `other-options“`
  - Run only the g++ “old-deja” tests in the testsuite with filenames matching ‘9805*’: `make check-g++ RUNTESTFLAGS="old-deja.exp=9805* other-options"`

[1] refre to section testsuite organization. The most important ones are compiler.exp, execute.exp, dg.exp and old-deja.exp.

Running GCC testsuite (3/5)

share/dejagnu/baseboards:
Arm-sim.exp
I386-sid.exp
Linux-gdbserver.exp
Mips-sim-idt64.exp
Riscv-sim.exp

- Passing options of target boards and compiler options:

```
make check-g++ RUNTESTFLAGS="--target_board=unix/-O3/-fmerge-constants"
```

- Run multiple times using combinations of options:

```
…"--target_board=arm-sim\{-mhard-float,-msoft-float\}\{-O1,-O2,-O3,\}"
```

As if

```
--target_board='arm-sim/-mhard-float/-O1 \
                arm-sim/-mhard-float/-O2 \
                arm-sim/-mhard-float/-O3 \
                arm-sim/-mhard-float \
                arm-sim/-msoft-float/-O1 \
                arm-sim/-msoft-float/-O2 \
                arm-sim/-msoft-float/-O3 \
                arm-sim/-msoft-float'
```

Serial→Parallel

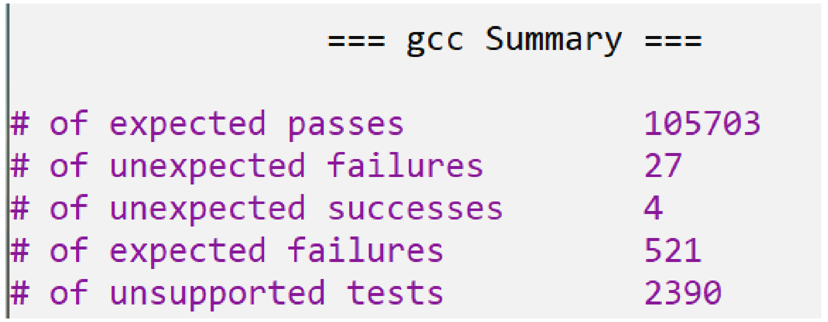
```
make -j4 check-gcc//arm-sim/{-mhard-float,-msoft-float}/{-O1,-O2,-O3,}
```

Running GCC testsuite (4/5)

- How to interpret test results

PASS	the test passed as expected
XPASS	the test unexpectedly passed
FAIL	the test unexpectedly failed
XFAIL	the test failed as expected
UNSUPPORTED	the test is not supported on this platform
ERROR	the testsuite detected an error
WARNING	the testsuite detected a possible problem

- Compare your results



```
                === gcc Summary ===

# of expected passes            105703
# of unexpected failures        27
# of unexpected successes       4
# of expected failures          521
# of unsupported tests          2390
```

Running GCC testsuite (5/5)

- How to test GCC on a simulator
  - Use a combined tree (not required, but convenient. Contains binutils, newlib, sim)
  - Configure,Make and Test:

```
configure --target=arm-eabi --enable-languages=c,c++ --prefix=INSTALLDIR
make
make check RUNTESTFLAGS= --target_board=arm-sim[1]
```

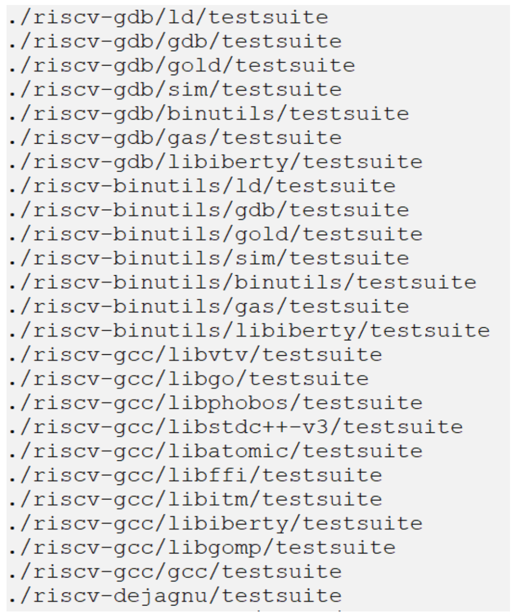
[1] find out the name of the target board is to look in the DejaGNU sources, in /usr/share/dejagnu/baseboards

Content

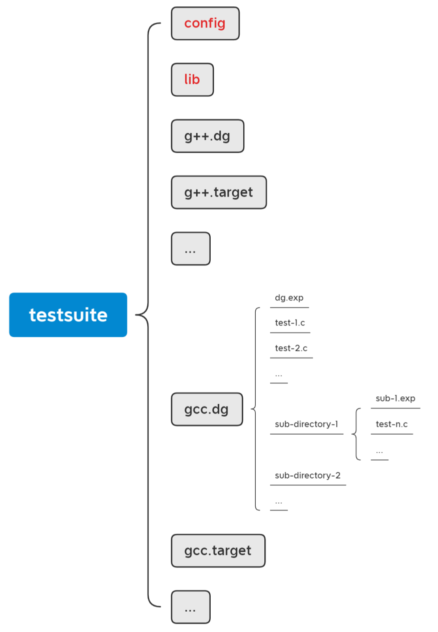
- Running GCC testsuite
    - Testing a cross-compiler
- **Testsuite organization**
- Adding new tests

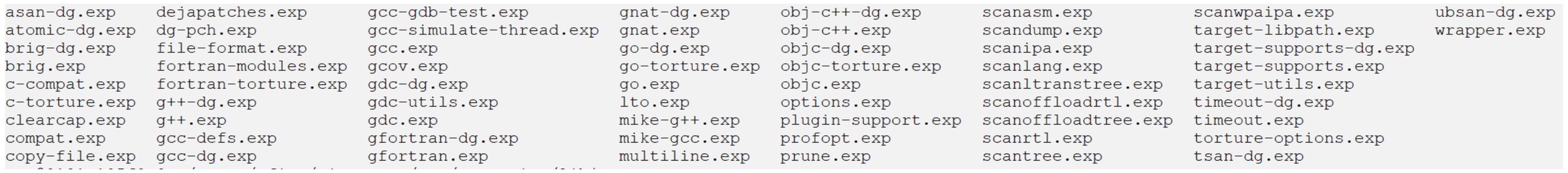
Testsuite organization (1/5)

- Each tool or module has its own testsuite directory
  - How to run these testsuites: refer to their build guide or Makefile
- Tree structure
  - Config: a default.exp for unsupported targets
  - lib: DejaGnu will load these tool specific configuration files. For a tool named *toolname*, the file will be lib/*toolname*.exp



```
./riscv-gdb/ld/testsuite
./riscv-gdb/gdb/testsuite
./riscv-gdb/gold/testsuite
./riscv-gdb/sim/testsuite
./riscv-gdb/binutils/testsuite
./riscv-gdb/gas/testsuite
./riscv-gdb/libiberty/testsuite
./riscv-binutils/ld/testsuite
./riscv-binutils/gdb/testsuite
./riscv-binutils/gold/testsuite
./riscv-binutils/sim/testsuite
./riscv-binutils/binutils/testsuite
./riscv-binutils/gas/testsuite
./riscv-binutils/libiberty/testsuite
./riscv-gcc/libvtv/testsuite
./riscv-gcc/libgo/testsuite
./riscv-gcc/libphobos/testsuite
./riscv-gcc/libstdc++-v3/testsuite
./riscv-gcc/libatomic/testsuite
./riscv-gcc/libffi/testsuite
./riscv-gcc/libitm/testsuite
./riscv-gcc/libiberty/testsuite
./riscv-gcc/libgomp/testsuite
./riscv-gcc/gcc/testsuite
./riscv-dejagnu/testsuite
```



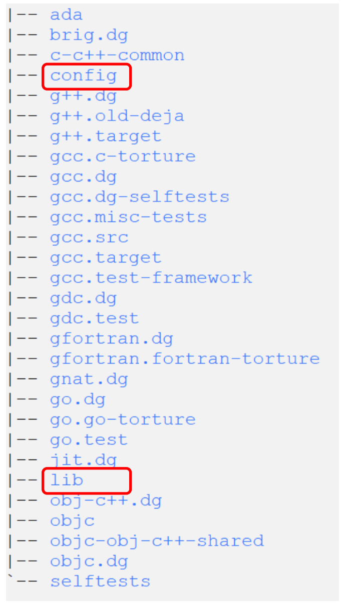


```
asan-dg.exp      dejapatches.exp      gcc-gdb-test.exp          gnat-dg.exp      obj-c++-dg.exp      scanasm.exp           scanwpaipa.exp            ubsan-dg.exp
atomic-dg.exp    dg-pch.exp           gcc-simulate-thread.exp   gnat.exp         obj-c++.exp         scandump.exp          target-libpath.exp        wrapper.exp
brig-dg.exp      file-format.exp      gcc.exp                   go-dg.exp        objc-dg.exp         scanipa.exp           target-supports-dg.exp
brig.exp         fortran-modules.exp  gcov.exp                  go-torture.exp   objc-torture.exp    scanlang.exp          target-supports.exp
c-compat.exp     fortran-torture.exp  gdc-dg.exp                go.exp           objc.exp            scanltranstree.exp    target-utils.exp
c-torture.exp    g++-dg.exp           gdc-utils.exp             lto.exp          options.exp         scanoffloadrtl.exp    timeout-dg.exp
clearcap.exp     g++.exp              gdc.exp                   mike-g++.exp     plugin-support.exp  scanoffloadtree.exp   timeout.exp
compat.exp       gcc-defs.exp         gfortran-dg.exp           mike-gcc.exp     profopt.exp         scanrtl.exp           torture-options.exp
copy-file.exp    gcc-dg.exp           gfortran.exp              multiline.exp    prune.exp           scantree.exp          tsan-dg.exp
```

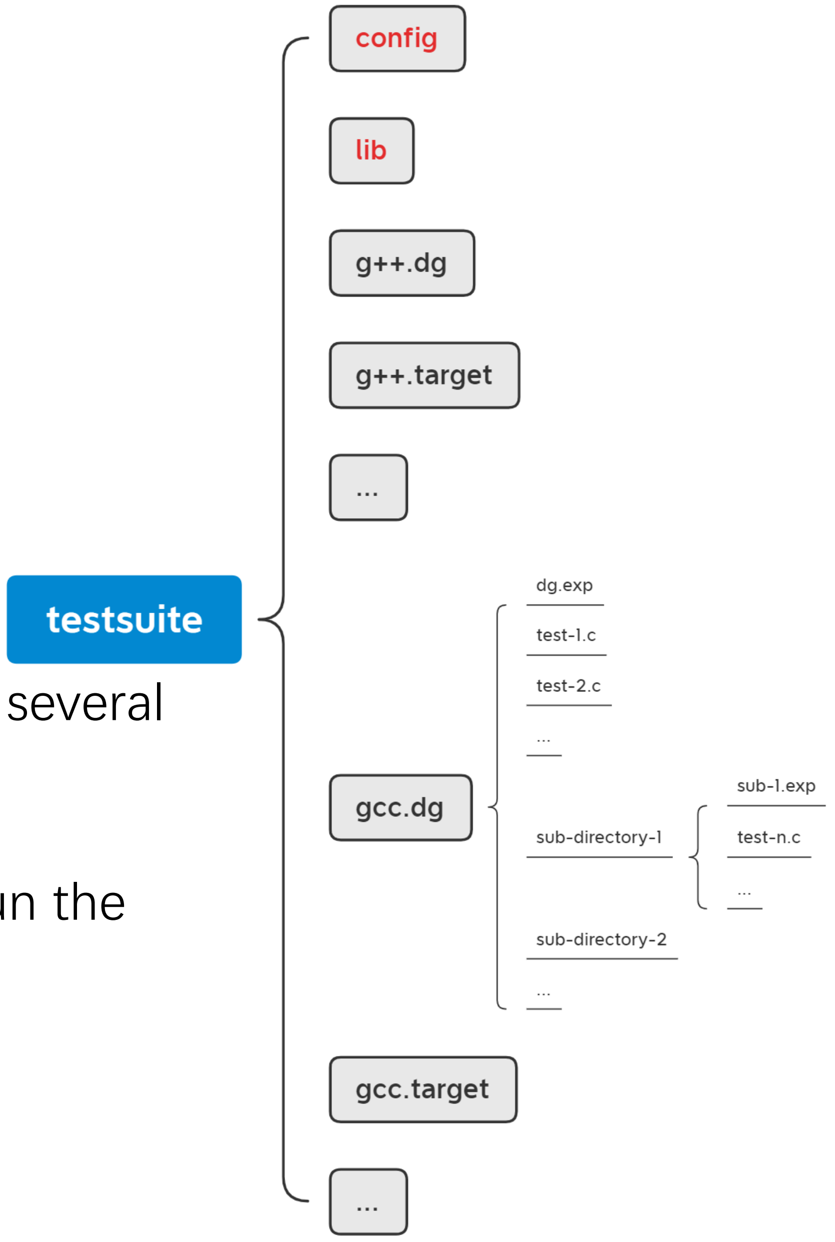
Testsuite organization (2/5)

- Tree structure
  - Config
  - Lib
  - Test directories
    - Directories name: *toolname*[type][.tests]
    - The optional type field allows the tests to be split into several directories. eg. gcc.dg, gcc.target
    - Test directory = .exp file + testcase + sub-dir
    - Every test directory should have at least one .exp to run the testcases.

make check-*toolname*

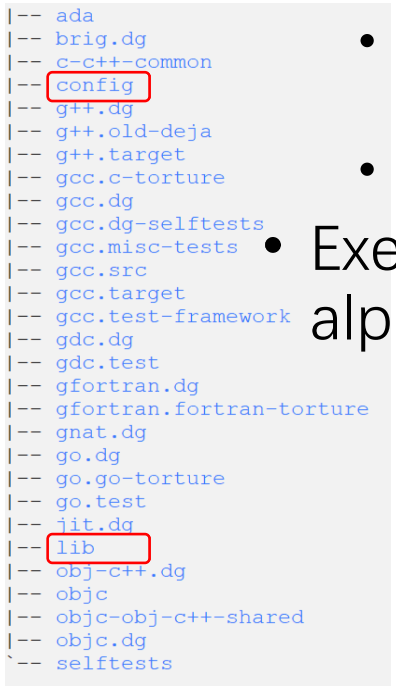


```
|-- ada
|-- brig.dg
|-- c-c++-common
|-- config
|-- g++.dg
|-- g++.old-deja
|-- g++.target
|-- gcc.c-torture
|-- gcc.dg
|-- gcc.dg-selftests
|-- gcc.misc-tests
|-- gcc.src
|-- gcc.target
|-- gcc.test-framework
|-- gdc.dg
|-- gdc.test
|-- gfortran.dg
|-- gfortran.fortran-torture
|-- gnat.dg
|-- go.dg
|-- go.go-torture
|-- go.test
|-- jit.dg
|-- lib
|-- obj-c++.dg
|-- objc
|-- objc-obj-c++-shared
|-- objc.dg
`-- selftests
```

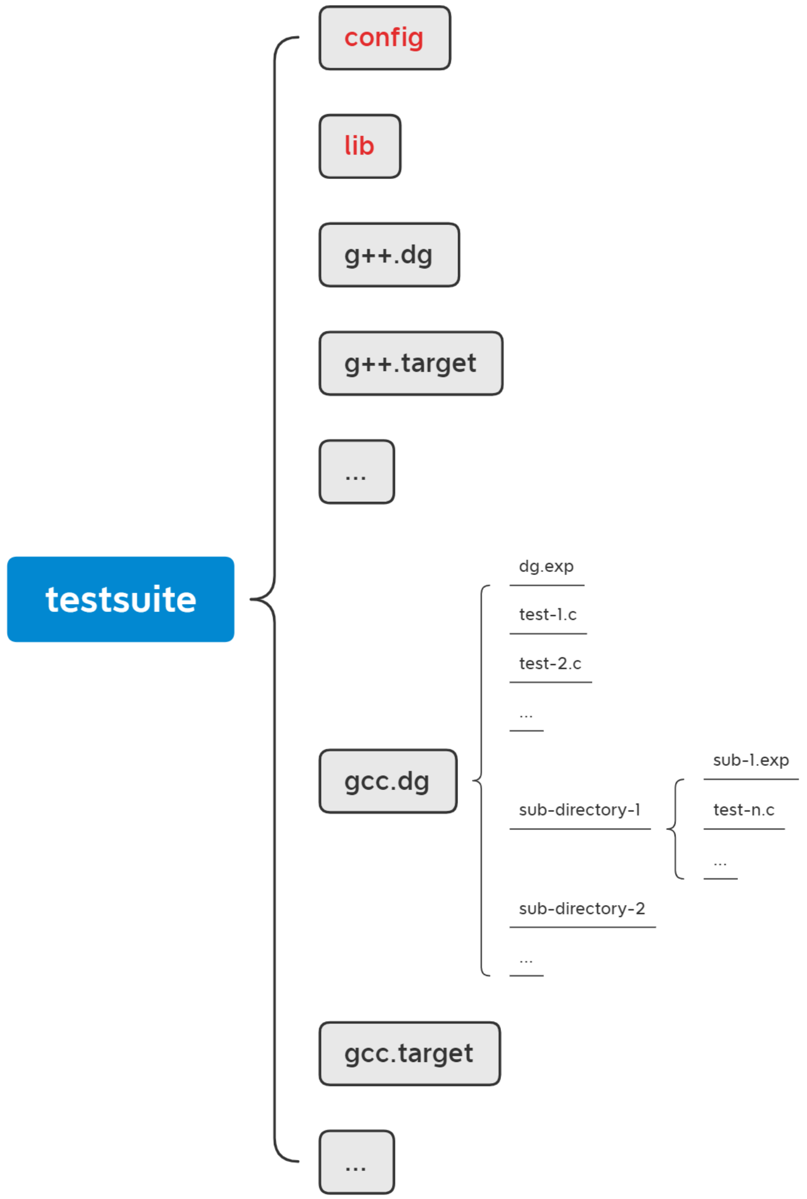


Testsuite organization (3/5)

- Execution Sequence
  - Which tool’s test to run?
    - configure --enable-language=c,c++
    - See makefile
    - make check-gcc; make check-g++, make check-gfortran
    - make check RUNTESTFLAGS=“--tool *toolname*”
  - Execution sequence: Iterate through folders alphabetically, search for .exp files, and execute.

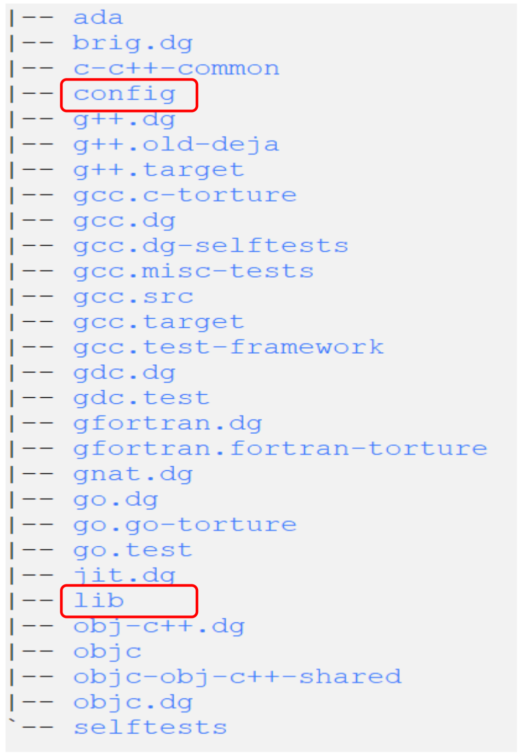


```
|-- ada
|-- brig.dg
|-- c-c++-common
|-- config
|-- g++.dg
|-- g++.old-deja
|-- g++.target
|-- gcc.c-torture
|-- gcc.dg
|-- gcc.dg-selftests
|-- gcc.misc-tests
|-- gcc.src
|-- gcc.target
|-- gcc.test-framework
|-- gdc.dg
|-- gdc.test
|-- gfortran.dg
|-- gfortran.fortran-torture
|-- gnat.dg
|-- go.dg
|-- go.go-torture
|-- go.test
|-- jit.dg
|-- lib
|-- obj-c++.dg
|-- objc
|-- objc-obj-c++-shared
|-- objc.dg
`-- selftests
```



Testsuite organization (4/5)

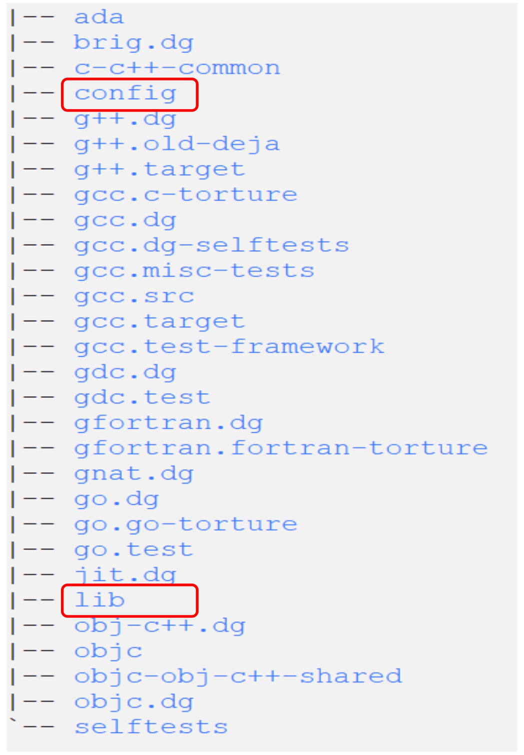
- GCC testsuite
  - ada: the Ada language testsuites.
  - brig.dg: BRIG(HSALL) frontend test case.
  - c-c++-common: for both c and c++ test
  - g++.dg: All new G++ tests should be placed here.
  - g++.target/gcc.target: test for different processor architecture.
  - gcc.dg: correctness tests for various compiler features should go here if possible.
  - gcc.c-torture: contains code fragments broken easily historically. Run with multiple optimization options, so tests for features which only break at some optimization levels belong here.



```
|-- ada
|-- brig.dg
|-- c-c++-common
|-- config
|-- g++.dg
|-- g++.old-deja
|-- g++.target
|-- gcc.c-torture
|-- gcc.dg
|-- gcc.dg-selftests
|-- gcc.misc-tests
|-- gcc.src
|-- gcc.target
|-- gcc.test-framework
|-- gdc.dg
|-- gdc.test
|-- gfortran.dg
|-- gfortran.fortran-torture
|-- gnat.dg
|-- go.dg
|-- go.go-torture
|-- go.test
|-- jit.dg
|-- lib
|-- obj-c++.dg
|-- objc
|-- objc-obj-c++-shared
|-- objc.dg
`-- selftests
```

Testsuite organization (5/5)

- GCC testsuite
    - gcc.dg-selftests: Tests that test dejagnu extensions used in gcc testing.
    - gcc.misc-tests: miscellaneous tests
    - gcc.src: ??
    - gcc.test-framework: check the test directives used in GCC’s testsuite.
    - gdc.dg: GCC testsuite for D programming language
    - gdc.test: D2 programming language testsuite import from DMD compiler hosted at https://github.com/dlang/dmd/.
    - gfortran.dg: GCC testsuite for fortran
    - gnat.gd: test for Ada compiler
    - go.dg: GCC testsuite for go language
    - go.test: Test using the testsuite for the gc Go compiler.
    - jit.dg: Test code for libgccjit.so
    - obj-c++.dg: GCC testsuite for obj-c++
    - objc.dg: GCC testsuite for objc



```
|-- ada
|-- brig.dg
|-- c-c++-common
|-- config
|-- g++.dg
|-- g++.old-deja
|-- g++.target
|-- gcc.c-torture
|-- gcc.dg
|-- gcc.dg-selftests
|-- gcc.misc-tests
|-- gcc.src
|-- gcc.target
|-- gcc.test-framework
|-- gdc.dg
|-- gdc.test
|-- gfortran.dg
|-- gfortran.fortran-torture
|-- gnat.dg
|-- go.dg
|-- go.go-torture
|-- go.test
|-- jit.dg
|-- lib
|-- obj-c++.dg
|-- objc
|-- objc-obj-c++-shared
|-- objc.dg
`-- selftests
```

Content

- Running GCC testsuite
  - Testing a cross-compiler
- Testsuite organization
- **Adding new tests**

Adding new test (1/3)

- Two ways
  - Only add test case in existed folder
  - Create your own test directory

    (create both .exp driver and testcase)

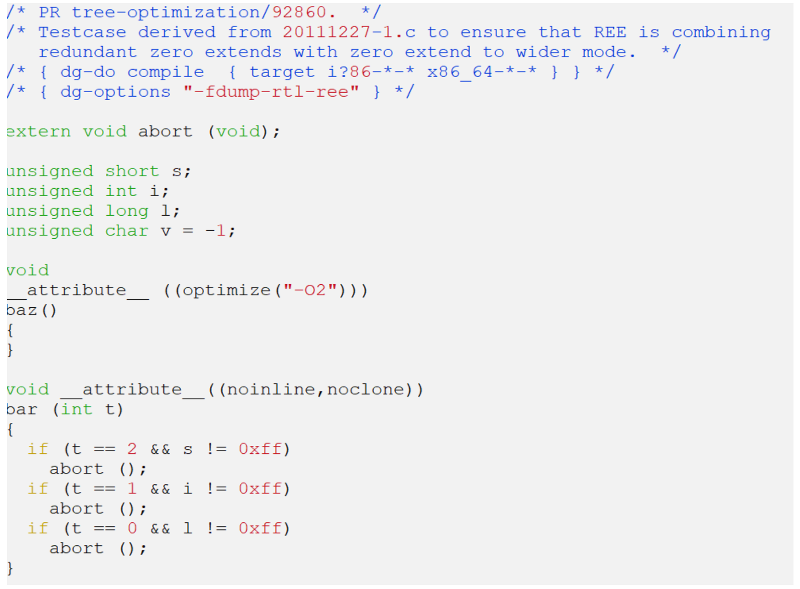
- Add .exp test driver

```
load_lib ${tool}-dg.exp
dg-init
dg-runtest [lsort [glob –nocomplain $srcdir/$subdir/foo*]] ...
dg-finish
```

Adding new test (2/3)

- Add testcase
  - Test Directives:

{ dg-do *do-what-keyword* [{ target/xfail *selector* }] }	**Specify how to build the test** *do-what-keyword* specifies how the test is compiled and whether it is executed. It is one of: preprocess, compile, assemble, link, run.
{ dg-options options [{ target selector }] }	**Specify additional compiler options** This DejaGnu directive provides a list of compiler options, to be used if the target system matches *selector*, that replace the default options used for this set of tests.
{ dg-skip-if comment { selector } [{ include-opts } [{ exclude-opts }]] }	**Skip a test for some targets** Skip the test if all of the following conditions are met. To skip a test if either -O2 or -O3 is used with -g but not if -fpic is also present: /* { dg-skip-if "" { *-*-* } { "-O2 -g" "-O3 -g" } { "-fpic" } } */



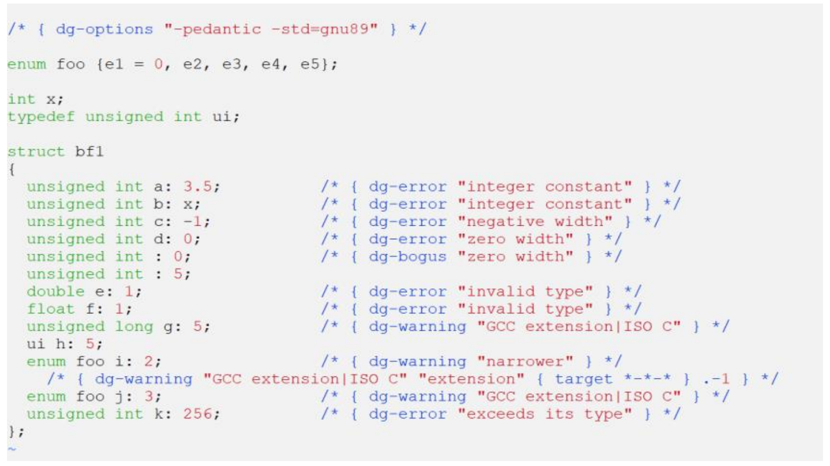
```c
/* PR tree-optimization/92860.  */
/* Testcase derived from 20111227-1.c to ensure that REE is combining
   redundant zero extends with zero extend to wider mode.  */
/* { dg-do compile  { target i?86-*-* x86_64-*-* } } */
/* { dg-options "-fdump-rtl-ree" } */

extern void abort (void);

unsigned short s;
unsigned int i;
unsigned long l;
unsigned char v = -1;

void
__attribute__ ((optimize("-O2")))
baz()
{
}

void __attribute__((noinline,noclone))
bar (int t)
{
  if (t == 2 && s != 0xff)
    abort ();
  if (t == 1 && i != 0xff)
    abort ();
  if (t == 0 && l != 0xff)
    abort ();
}
```



```c
/* { dg-options "-pedantic -std=gnu89" } */

enum foo {e1 = 0, e2, e3, e4, e5};

int x;
typedef unsigned int ui;

struct bf1
{
  unsigned int a: 3.5;            /* { dg-error "integer constant" } */
  unsigned int b: x;              /* { dg-error "integer constant" } */
  unsigned int c: -1;             /* { dg-error "negative width" } */
  unsigned int d: 0;              /* { dg-error "zero width" } */
  unsigned int : 0;               /* { dg-bogus "zero width" } */
  unsigned int : 5;
  double e: 1;                    /* { dg-error "invalid type" } */
  float f: 1;                     /* { dg-error "invalid type" } */
  unsigned long g: 5;             /* { dg-warning "GCC extension|ISO C" } */
  ui h: 5;
  enum foo i: 2;                  /* { dg-warning "narrower" } */
    /* { dg-warning "GCC extension|ISO C" "extension" { target *-*-* } .-1 } */
  enum foo j: 3;                  /* { dg-warning "GCC extension|ISO C" } */
  unsigned int k: 256;            /* { dg-error "exceeds its type" } */
};
~
```

Adding new test (3/3)

- Add testcase
  - Test Directives[1]:

{ dg-xfail-if *comment* { *selector* } [{ *include-opts* } [{ *exclude-opts* }]] }	**Expect a test to fail for some targets** Expect the test to fail if the conditions (which are the same as for **dg-skip-if**) are met. This does not affect the execute step.
{ dg-error *regexp* [*comment* [{ target/xfail *selector* } [*line*]]] }	**Verify compiler messages** This DejaGnu directive appears on a source line that is expected to get an error message, or else specifies the source line associated with the message. If there is no message for that line or if the text of that message is not matched by *regexp* then the check fails and *comment* is included in the FAIL message. The check does not look for the string 'error' unless it is part of *regexp*.
{ dg-output *regexp* [{ target/xfail *selector* }] }	**Verify output of the test executable** This DejaGnu directive compares *regexp* to the combined output that the test executable writes to stdout and stderr.
{ dg-final { *local-directive* } }	**Add checks at the end of a test** This DejaGnu directive is placed within a comment anywhere in the source file and is processed after the test has been compiled and run. Multiple 'dg-final' commands are processed in the order in which they appear in the source file. See Final Actions, for a list of directives that can be used within dg-final.

[1] refer to $INSTALLDIR/share/dejagnu/dg.exp and gcc/testsuite/lib

THE END

Thanks For Watching