GoogleTest

This module defines functions to help use the Google Test infrastructure. Two mechanisms for adding tests are provided. gtest_add_tests() has been around for some time, originally via find_package(GTest). gtest_discover_tests() was introduced in CMake 3.10.

The (older) gtest_add_tests() scans source files to identify tests. This is usually effective, with some caveats, including in cross-compiling environments, and makes setting additional properties on tests more convenient. However, its handling of parameterized tests is less comprehensive, and it requires re-running CMake to detect changes to the list of tests.

The (newer) gtest_discover_tests() discovers tests by asking the compiled test executable to enumerate its tests. This is more robust and provides better handling of parameterized tests, and does not require CMake to be re-run when tests change. However, it may not work in a cross-compiling environment, and setting test properties is less convenient.

More details can be found in the documentation of the respective functions.

Both commands are intended to replace use of add_test() to register tests, and will create a separate CTest test for each Google Test test case. Note that this is in some cases less efficient, as common set-up and tear-down logic cannot be shared by multiple test cases executing in the same instance. However, it provides more fine-grained pass/fail information to CTest, which is usually considered as more beneficial. By default, the CTest test name is the same as the Google Test name (i.e. suite.testcase); see also TEST_PREFIX and TEST_SUFFIX.

gtest_add_tests

Automatically add tests with CTest by scanning source code for Google Test macros:

```
gtest_add_tests(TARGET target
                [SOURCES src1...]
                [EXTRA_ARGS arg1...]
                [WORKING_DIRECTORY dir]
                [TEST_PREFIX prefix]
                [TEST_SUFFIX suffix]
                [SKIP_DEPENDENCY]
                [TEST_LIST outVar]
)
```

gtest_add_tests attempts to identify tests by scanning source files. Although this is generally effective, it uses only a basic regular expression match, which can be defeated by atypical test declarations, and is unable to fully "split" parameterized tests. Additionally, it requires that CMake be re-run to discover any newly added, removed or renamed tests (by default, this means that CMake is re-run when any test source file is changed, but see SKIP_DEPENDENCY). However, it has the advantage of declaring tests at CMake time, which somewhat simplifies setting additional properties on tests, and always works in a cross-compiling environment.

The options are:

TARGET target

Specifies the Google Test executable, which must be a known CMake executable target. CMake will substitute the location of the built executable when running the test.

SOURCES src1...

When provided, only the listed files will be scanned for test cases. If this option is not given, the SOURCES property of the specified target will be used to obtain the list of sources.

EXTRA_ARGS arg1...

Any extra arguments to pass on the command line to each test case.

WORKING_DIRECTORY dir

Specifies the directory in which to run the discovered test cases. If this option is not provided, the current binary directory is used.

TEST_PREFIX prefix

Specifies a prefix to be prepended to the name of each discovered test case. This can be useful when the same source files are being used in multiple calls to gtest_add_test() but with different EXTRA_ARGS.

TEST_SUFFIX suffix

Similar to TEST_PREFIX except the suffix is appended to the name of every discovered test case. Both TEST_PREFIX and TEST_SUFFIX may be specified.

SKIP_DEPENDENCY

Normally, the function creates a dependency which will cause CMake to be re-run if any of the sources being scanned are changed. This is to ensure that the list of discovered tests is updated. If this behavior is not desired (as may be the case while actually writing the test cases), this option can be used to prevent the dependency from being added.

TEST_LIST outVar

The variable named by outVar will be populated in the calling scope with the list of discovered test cases. This allows the caller to do things like manipulate test properties of the discovered tests.

```
include(GoogleTest)
add_executable(FooTest FooUnitTest.cxx)
gtest_add_tests(TARGET      FooTest
                TEST_SUFFIX .noArgs
                TEST_LIST   noArgsTests
)
gtest_add_tests(TARGET      FooTest
                EXTRA_ARGS  --someArg someValue
                TEST_SUFFIX .withArgs
                TEST_LIST   withArgsTests
)
set_tests_properties(${noArgsTests}   PROPERTIES TIMEOUT 10)
set_tests_properties(${withArgsTests} PROPERTIES TIMEOUT 20)
```

For backward compatibility, the following form is also supported:

```
gtest_add_tests(exe args files...)
```

exe

The path to the test executable or the name of a CMake target.

args

A ;-list of extra arguments to be passed to executable. The entire list must be passed as a single argument. Enclose it in quotes, or pass "" for no arguments.

files...

A list of source files to search for tests and test fixtures. Alternatively, use AUTO to specify that exe is the name of a CMake executable target whose sources should be scanned.

```
include(GoogleTest)
set(FooTestArgs --foo 1 --bar 2)
add_executable(FooTest FooUnitTest.cxx)
gtest_add_tests(FooTest "${FooTestArgs}" AUTO)
```

gtest_discover_tests

Automatically add tests with CTest by querying the compiled test executable for available tests:

```
gtest_discover_tests(target
                     [EXTRA_ARGS arg1...]
                     [WORKING_DIRECTORY dir]
                     [TEST_PREFIX prefix]
                     [TEST_SUFFIX suffix]
                     [NO_PRETTY_TYPES] [NO_PRETTY_VALUES]
                     [PROPERTIES name1 value1...]
                     [TEST_LIST var]
)
```

gtest_discover_tests sets up a post-build command on the test executable that generates the list of tests by parsing the output from running the test with the --gtest_list_tests argument. Compared to the source parsing approach of gtest_add_tests(), this ensures that the full list of tests, including instantiations of parameterized tests, is obtained. Since test discovery occurs at build time, it is not necessary to re-run CMake when the list of tests changes. However, it requires that CROSSCOMPILING_EMULATOR is properly set in order to function in a cross-compiling environment.

Additionally, setting properties on tests is somewhat less convenient, since the tests are not available at CMake time. Additional test properties may be assigned to the set of tests as a whole using the PROPERTIES option. If more fine-grained test control is needed, custom content may be provided through an external CTest script using the TEST_INCLUDE_FILES directory property. The set of discovered tests is made accessible to such a script via the <target>_TESTS variable.

The options are:

target

Specifies the Google Test executable, which must be a known CMake executable target. CMake will substitute the location of the built executable when running the test.

EXTRA_ARGS arg1...

Any extra arguments to pass on the command line to each test case.

WORKING_DIRECTORY dir

Specifies the directory in which to run the discovered test cases. If this option is not provided, the current binary directory is used.

TEST_PREFIX prefix

Specifies a prefix to be prepended to the name of each discovered test case. This can be useful when the same test executable is being used in multiple calls to gtest_discover_tests() but with different EXTRA_ARGS.

TEST_SUFFIX suffix

Similar to TEST_PREFIX except the suffix is appended to the name of every discovered test case. Both TEST_PREFIX and TEST_SUFFIX may be specified.

NO_PRETTY_TYPES

By default, the type index of type-parameterized tests is replaced by the actual type name in the CTest test name. If this behavior is undesirable (e.g. because the type names are unwieldy), this option will suppress this behavior.

NO_PRETTY_VALUES

By default, the value index of value-parameterized tests is replaced by the actual value in the CTest test name. If this behavior is undesirable (e.g. because the value strings are unwieldy), this option will suppress this behavior.

PROPERTIES name1 value1...

Specifies additional properties to be set on all tests discovered by this invocation of gtest_discover_tests.

TEST_LIST var

Make the list of tests available in the variable var, rather than the default <target>_TESTS. This can be useful when the same test executable is being used in multiple calls to gtest_discover_tests(). Note that this variable is only available in CTest.

TIMEOUT num

Specifies how long (in seconds) CMake will wait for the test to enumerate available tests. If the test takes longer than this, discovery (and your build) will fail. Most test executables will enumerate their tests very quickly, but under some exceptional circumstances, a test may require a longer timeout. The default is 5. See also the TIMEOUT option of execute_process().