Can I pass parameters to googletest test function

Ask Question

After building my testfile, xxxxtest, with gtest can I pass a parameter when running the test, e.g. `./xxxxtest 100` . I want to control my test function using the parameter, but I do not know how to use the para in my test, can you show me a sample in test?

c++ googletest

edited Mar 1 '12 at 10:29 Chris 5,064 1 28 48

asked Feb 29 '12 at 9:42 gino 60 2 9

1 possible duplicate of How to pass parameters to the gtest – Rob Kennedy Mar 19 '12 at 22:25

2 Answers

You could do something like the following:

main.cc

```
#include <string>
#include "gtest/gtest.h"
#include "my_test.h"

int main(int argc, char **argv) {
  std::string command_line_arg(argc == 2 ? argv[1] : "");
  testing::InitGoogleTest(&argc, argv);
  testing::AddGlobalTestEnvironment(new MyTestEnvironment(command_line_arg));
  return RUN_ALL_TESTS();
}
```

my_test.h

```
#include <string>
#include "gtest/gtest.h"

namespace {
std::string g_command_line_arg;
}

class MyTestEnvironment : public testing::Environment {
 public:
  explicit MyTestEnvironment(const std::string &command_line_arg) {
    g_command_line_arg = command_line_arg;
  }
};

TEST(MyTest, command_line_arg_test) {
  ASSERT_FALSE(g_command_line_arg.empty());
}
```

answered Mar 1 '12 at 3:20 Fraser 46.7k 6 147 166

What's the purpose of the `Environment` descendant? Why not just `g_command_line_arg = argc == 2 ? argv[1] : ""` ? – Rob Kennedy Mar 19 '12 at 22:25

It's just to limit the scope of `g_command_line_arg` . Since it's in an unnamed namespace in my_test.h, it's not accessible outside of that translation unit. – Fraser Mar 20 '12 at 8:09

Why does it give me an error of `free(): invalid pointer <a_number>` ? – thedarkside ofthemoon Jul 16 '14 at 15:11

Shouldn't you the inialization of `command_line_arg` after the `InitGoogleTest` , so you don't interfere with parameters to gtest itself? – Philippos Nov 20 '17 at 10:09

You should be using Type-Parameterized Tests.
https://code.google.com/p/googletest/wiki/AdvancedGuide#Type-Parameterized_Tests

> Type-parameterized tests are like typed tests, except that they don't require you to know the list of types ahead of time. Instead, you can define the test logic first and instantiate it with different type lists later. You can even instantiate it more than once in the same program.
>
> If you are designing an interface or concept, you can define a suite of type-parameterized tests to verify properties that any valid implementation of the interface/concept should have. Then, the author of each implementation can just instantiate the test suite with his type to verify that it conforms to the requirements, without having to write similar tests repeatedly.

Example

```
class FooTest: public ::testing::TestWithParam < int >{....};
    TEST_P(FooTest, DoesBar){
        ASSERT_TRUE(foo.DoesBar(GetParam()));
    }

INSTANTIATE_TEST_CASE_P(OneToTenRange, FooTest, ::testing::Range(1, 10));
```

answered Sep 15 '14 at 17:48
Basanta
111 6