Specifying output string argument with GoogleMock

Ask Question

I'm evaluating the Google Test/Mock as a framework for unit tests of my C code.

How can I specify the output string argument for the function I would like to mock?

Here I have `int get_int_param(const char *)` is the function to test and it uses `int _get_text_file_content(const char *fn, char *content)` function that I want to mock.

How to specify this `char *content` that is going to be the result of execution of mocking function?

I'm struggling with this code:

```
TEST(GetParameterTest,Positiv){
   const static int strLen=29;
   char *text=(char *)calloc(strLen,1);
   strcpy(text, "param1=1\nparam2=42\nparam3=3");

   MokedFunctions mokedFunctions;
   EXPECT_CALL(mokedFunctions,
_get_text_file_content("process.conf",_)).Times(AtLeast(1)).WillOnce(SetArgReferee<1>
(text));

   EXPECT_EQ(1, get_int_param("param1"));
}
```

and got this compile error:

```
/usr/include/gmock/gmock-more-actions.h: In instantiation of ‘typename
testing::internal::Function<F>::Result testing::SetArgRefereeActionP<k,
value_type>::gmock_Impl<F>::gmock_PerformImpl(const args_type&,
arg0_type, arg1_type, arg2_type, arg3_type, arg4_type, arg5_type,
arg6_type, arg7_type, arg8_type, arg9_type) const [with arg0_type =
const char*; arg1_type = char*; arg2_type =
testing::internal::ExcessiveArg; arg3_type =
testing::internal::ExcessiveArg; arg4_type =
testing::internal::ExcessiveArg; arg5_type =
testing::internal::ExcessiveArg; arg6_type =
testing::internal::ExcessiveArg; arg7_type =
testing::internal::ExcessiveArg; arg8_type =
testing::internal::ExcessiveArg; arg9_type =
testing::internal::ExcessiveArg; F = int(const char*, char*); int k = 1;
value_type = char*; typename testing::internal::Function<F>::Result =
int; testing::SetArgRefereeActionP<k,
value_type>::gmock_Impl<F>::args_type = std::tuple<const char*, char*>]’:
```

```
/usr/include/gmock/gmock-generated-actions.h:664:23:   required from
'static Result testing::internal::ActionHelper<Result,
Impl>::Perform(Impl*, const std::tuple<_U1, _U2>&) [with A0 = const
char*; A1 = char*; Result = int; Impl =
testing::SetArgRefereeActionP<1, char*>::gmock_Impl<int(const char*,
char*)>]'

/usr/include/gmock/gmock-more-actions.h:168:1:   required from
'testing::SetArgRefereeActionP<k,
value_type>::gmock_Impl<F>::return_type
testing::SetArgRefereeActionP<k,
value_type>::gmock_Impl<F>::Perform(const args_type&) [with F =
int(const char*, char*); int k = 1; value_type = char*;
testing::SetArgRefereeActionP<k,
value_type>::gmock_Impl<F>::return_type = int;
testing::SetArgRefereeActionP<k, value_type>::gmock_Impl<F>::args_type
= std::tuple<const char*, char*>]'

test_param.cpp:68:1:   required from here
/usr/include/gmock/gmock-more-actions.h:175:3: error: size of array is negative
GTEST_COMPILE_ASSERT_(internal::is_reference<argk_type>::value,
^
In file included from /usr/include/gmock/gmock.h:65:0,
            from test_param.cpp:2:
 /usr/include/gmock/gmock-more-actions.h:177:28: error: assignment of read-only location
'std::get<1u, {const char*, char*}>((* & args))'
    ::std::tr1::get<k>(args) = value;
                           ^
 make[1]: *** [test_param.o] Error 1
```

What I'm doing wrong?

c unit-testing googletest googlemock

asked May 12 '15 at 9:09
evrdrkgrn0
46 3

2 Answers

`SetArgReferee` expects the argument to be a C++ reference, which it's not it your case.

In general, in order to better understand these actions it helps to think of them as operations over the argument `arg`:

- `SetArgPointee(value)` is essentially `*arg = value` (`arg` must be a pointer)
- `SetArgReferee(value)` is `arg = value` (`arg` must be a reference)
- `SetArrayArgument(first, last)` is `memcpy(arg, first, last - first)` (`arg` must be a pointer)
- `SaveArg(ptr)` is `*ptr = arg`
- `SaveArgPointee(ptr)` is `*ptr = *arg` (`arg` must be a pointer)

Given that, it becomes obvious that the action you need is `SetArrayArgument<1>(text, text + strlen(text) + 1)` .

answered May 13 '15 at 4:03
VladLosev
4,411 1 16 36

You are using a C++ testing framework for C code. Unless you're willing to compile your code with g++ (instead of gcc), it's not going to work. Gcc can't compile C++ code.

answered May 12 '15 at 9:13
rost0031
1,710 9 19

Yes sure, I'm compiling my C code with gcc and then compile my tests and link them with g++. No problem, it is working quite good. – evrdrkgrn0 May 12 '15 at 9:17

Ok, so are you doing something like this? stackoverflow.com/questions/19552805/… – rost0031 May 12 '15 at 9:22

yep, my solution is a bit dirty but it is based on this post – evrdrkgrn0 May 12 '15 at 9:33

In line EXPECT_CALL(mokedFunctions, _get_text_file_content("process.conf", _)).Times(AtLeast(1)).WillOnce(SetArgReferee<1>(text)); you have an underscore as an argument. I'm not an expert at google mock so I could be missing something but that doesn't look right. – rost0031 May 12 '15 at 9:41

`::testing::_` is a wildcard from GMock. It means that there is any value of correct type is expected. My understanding of this is: "I'm expecting everything that is correct pointer as an input" and then I want to specify the output at this pointer, that in turn is predefined (`SetArgReferee`). – evrdrkgrn0 May 12 '15 at 9:48