make test

m-nishi

VimConf 2019, Nov 3rd 2019
Have you ever...

built Vim?
- never: 68%
- a few: 17%
- a lot: 15%

run Vim's test?
- never: 85%
- a few: 15%
- a lot: 0%
About me

Name: m-nishi
Vim experience: 2 years
Embedded Software Engineer

Twitter: mnishz0
GitHub: mnishz
I will talk about...

• How to run Vim's test
• Overview
• Contents of test script
• Coverage
  • How I wrote a Vim's test
How to run Vim's test
How to run Vim's test

$ cd src && make test
   libtool-bin is required

--> Demo
How to run Vim's test

Result

Executed: 2198 Tests
Skipped: 35 Tests
FAILED: 1 Tests

Failures:
From test_assert.vim:
Found errors in Test_assert fail_function RunTheTest[40]..Test function RunTheTest[40]..Test_

---

Executed: 2201 Tests
Skipped: 35 Tests
Failed: 0 Tests

ALL DONE

TEST FAILURE
Overview

• Written in Vim script
• Also executed by CI services
• `:help testing` for details
Overview

• Written in Vim script
• Also executed by CI services
• `:help testing` for details
Overview

• Written in Vim script
• Also executed by CI services
• `:help testing` for details
Benefit of writing test

- Deep understanding of Vim
- Pull request may be merged quickly
- Contribution chance for the test!
Contents of test script
func SetUp()
    new dummy
    set nrformats&vim
endfunc

func TearDown()
    bwipe!
endfunc

func Test_visual_increment_01()
    call setline(1, repeat(["foobaar-10"], 5))

    call cursor(1, 1)
    exec "norm! <C-A>"
    call assert_equal("foobaar-9", getline('.'))
    call assert_equal([0, 1, 9, 0], getpos('.'))
**func** SetUp()
    new dummy
    set nrformats&vim
endfunc

**func** TearDown()
    bwipe!
endfunc

**func** Test_visual_increment_01()
    call setline(1, repeat(["foobaar-10"], 5))

    call cursor(1, 1)
    exec "norm! \<C-A>"
    call assert_equal("foobaar-9", getline('.'))
    call assert_equal([0, 1, 9, 0], getpos('.'))
func SetUp()
    new dummy
    set nrformats&vim
endfunc

func TearDown()
    bwipe!
endfunc

func Test_visual_increment_01()
    call setline(1, repeat(["foobaar-10"], 5))

    call cursor(1, 1)
    exec "norm! \<C-A>"
    call assert_equal("foobaar-9", getline('.'))
    call assert_equal([0, 1, 9, 0], getpos('.'))
func SetUp()
    new dummy
    set nrformats&vim
endfunc

func TearDown()
    bwipe!
endfunc

func Test_visual_increment_01()
    call setline(1, repeat(["foobaar-10"], 5))
    call cursor(1, 1)
    exec "norm! <C-A>"
    call assert_equal("foobaar-9", getline('.'))
    call assert_equal([0, 1, 9, 0], getpos('.'))
```vim
func SetUp()
    new dummy
    set nrformats&vim
endfunc

func TearDown()
    bwipe!
endfunc

func Test_visual_increment_01()
    call setline(1, repeat(["foobaar-10"], 5))
    call cursor(1, 1)
    exec "norm! <C-A>"
    call assert_equal("foobaar-9", getline('.'))
    call assert_equal([0, 1, 9, 0], getpos('.'))
```
```vim
func SetUp()
    new dummy
    set nrformats&vim
endfunc

func TearDown()
    bwipe!
endfunc

func Test_visual_increment_01()
    call setline(1, repeat(["foobaar-10"], 5))

    call cursor(1, 1)
    exec "norm! \<C-A>"
    call assert_equal("foobaar-9", getline('.'))
    call assert_equal([0, 1, 9, 0], getpos('.'))
```

src/testdir/test_increment.vim
terminal-diff

It compares

• window size
• text
• color
• other attributes
  • (bold, underline...)

```diff
1  hello there		 r one
2  another two
3  another three

----------------------
src/testdir/dumps/Test_popupwin_01.dump ===

bbbbbbbbbbbbbbbbbbbb

----------------------
src/testdir/failed/Test_popupwin_01.dump ===

1  hello there		 r one
2  another two
3  another three

1,1
```
Coverage
Coverage

• https://codecov.io/gh/vim/vim/vim
Coverage

- [https://codecov.io/gh/vim/vim](https://codecov.io/gh/vim/vim)
- Reached 80% on this May
  - Bram sent a thank-you message
- Pull request for test looks welcome.
How I wrote a Vim's test

1. Found not tested code
How I wrote a Vim's test

2. Studied how the code is executed

```cindent({lnum})

Get the amount of indent for line `{lnum}` according the C indenting rules, as with `cindent`.
The indent is counted in spaces, the value of `tabstop` is relevant. `{lnum}` is used just like in `getline()`.
When `{lnum}` is invalid or Vim was not compiled the `+cindent` feature, `-1` is returned.
See `C-indenting`.

Can also be used as a method:

```cindent
GetLnum() -> cindent()
```
How I wrote a Vim's test

3. Wrote a test script

```vim
+ func Test_cindent_func()
  + new
  + setlocal cindent
  + call setline(1, ['int main(void)', '{', 'return 0;', '}'])
  + call assert_equal(cindent(0), -1)
  + call assert_equal(cindent(3), &sw)
  + call assert_equal(cindent(line('.'+1)), -1)
  + bwipe!
  + endfunc
  +
" vim: shiftwidth=2 sts=2 expandtab
```
How I wrote a Vim's test

4. Checked coverage

```c
static void
f_cindent(ty whole T *argvars UNUSED, ty whole T *rettv)
{
    #ifdef FEAT_CINDENT
        pos T    pos;
        linenr T lnum;

    pos = curwin->w_cursor;
    lnum = tv_get_lnum(argvars);
    if (lnum >= 1 && lnum <= curbuf->bu_ml.ml_line_count)
    {
        curwin->w_cursor.lnum = lnum;
        rettv->vval.v_number = get_c_indent();
        curwin->w_cursor = pos;
    }
```
How I wrote a Vim's test

5. Sent a pull request!
Short summary

• $ make test
• Written by Vim script
• Let's write test for untested code!
• Thanks for flying with me!
How the test scripts are executed

$ vim -S runtest.vim test_increment.vim

```vim
func Setup()
....
endfunc

func Test_A()
....
endfunc
```

runtest.vim

中身どうしようか

```
interpret
execute
```
REDIR_TEST_TO_NULL

diff --git a/src/testdir/Makefile b/src/testdir/Makefile
index bcf2f8c37..0ffdbc228 100644
--- a/src/testdir/Makefile
+++ b/src/testdir/Makefile
@@ -12,7 +12,7 @@ SCRIPTSOURCE = .././runtime
    # Comment out this line to see the verbose output of tests.
    #
    # Catches SwapExists to avoid hanging at the ATTENTION prompt.
  -REDIR_TEST_TO_NULL = --cmd 'au SwapExists * let v:swapchoice = "e"' > /dev/null
  +REDIR_TEST_TO_NULL = --cmd 'au SwapExists * let v:swapchoice = "e"'
    
    # Uncomment this line to use valgrind for memory leaks and extra warnings.
    # The output goes into a file "valgrind.testN"
Test by CI services

**All checks have passed**

- LGTM analysis: JavaScript: Completed in 10m — No code changes detected
- LGTM analysis: Python: Completed in 484m — No code changes detected
- LGTM analysis: C/C++: Successful in 615m — No new or fixed alerts
- continuous-integration/appveyor/pr: AppVeyor build succeeded
- continuous-integration/travis ci/pr: The Travis CI build passed
- coverage/coveralls: Coverage increased (+0.1%) to 81.443%

**This branch has no conflicts with the base branch**

Only those with write access to this repository can merge pull requests.
Coverage, Codecov

What is Vim?
Coverage, Codecov
How to get test coverage

# TEST COVERAGE – Uncomment the two lines below the explanation to get code coverage information. (provided by Yegappan Lakshmanan)
# 1. Make clean, run configure and build Vim as usual.
# 2. Generate the baseline code coverage information:
#>-------$ lcov -c -i -b -d objects -o objects/coverage_base.info
# 3. Run "make test" to run the unit tests. The code coverage information will be generated in the src/objects directory.
# 4. Generate the code coverage information from the tests:
#>-------$ lcov -c -b -d objects/ -o objects/coverage_test.info
# 5. Combine the baseline and test code coverage data:
#>-------$ lcov -a objects/coverage_base.info -a objects/coverage_test.info -o obj
# 6. Process the test coverage data and generate a report in html:
#>-------$ genhtml objects/coverage_total.info -o objects
# 7. Open the objects/index.html file in a web browser to view the coverage information.
# PROFILE_CFLAGS=-g -O0 -fprofile-arcs -ftest-coverage
# LDFLAGS=---coverage
Makefiles

- `src/Makefile`
  - `unittests, test_libvterm, scripttests`
- `src/testdir/Makefile`
  - `.vim.res`
unittest

• # Unittest files

• JSON_TEST_SRC = json_test.c

• JSON_TEST_TARGET = json_test$(EXEEXT)

• KWORD_TEST_SRC = kword_test.c

• KWORD_TEST_TARGET = kword_test$(EXEEXT)

• MEMFILE_TEST_SRC = memfile_test.c

• MEMFILE_TEST_TARGET = memfile_test$(EXEEXT)

• MESSAGE_TEST_SRC = message_test.c

• MESSAGE_TEST_TARGET = message_test$(EXEEXT)
How to run single test

• Single test file
  • $ make test_****

• Single test or filtering
  • $ export TEST_FILTER=Test_****
How to create reference data for terminal dump

• Reference data are located in src/testdir/dumps
• term_dumpwrite()
Links

• http://rbttnn.hateblo.jp/entry/2019/03/24/054919
• https://daisuzu.hatenablog.com/entry/2017/12/02/174002
• https://qiita.com/m_nish/items/865751bedd1f7d28a740