



Speare Code Editor

The Small Perl IDE

For Perl Development


Copyright (C) 2020 Sevenuc Consulting
Version 1.0
Update: 4 Mar 2020

Speare (<http://sevenuc.com/en/Speare.html>) is an ultra lightweight code editor and a small IDE that has an efficient code navigation and call routines tracing ability, and has integrated debugging environment for C, C++ and Perl. It was originally developed to providing a native scripting language debugging environment that seamlessly integrated with C and C++. It not only has very concise user interface but also has a very flexible architecture to easily add a new programming language code runner, parser, syntax highlighting, code formatter and debugger in it.

For general Speare code editor usage, please refer this document:
http://sevenuc.com/download/Speare_quick_reference.pdf

Debug Mode

1. Show the debug toolbar

Click  siding bottom button.

2. Debug toolbar



From left to right, Start, Stop, Step Into, Step Out, Run To, Step Over, Show Watches.

The "Step Over" is equals to "Step next", and "Step To" is equals to "Continue" in common debugging words, and the "Step To" is the command that tell the debugger run to meet a breakpoint or an exception occurred or the program meet exit.

On the rightmost there are three other function units, they are, search items in the stackview, siding stackview, and clean the debug output.

Search in the debug output

Click in the output area and use the shortcut key "Control + F" to do the searching.

3. Socket Port

You can set the socket communication port number both used by Debug Server and the Speare code editor. Open the Preferences of Speare and select the "Debug Settings" tab then input your number.

Note: Please remember to empty the port number when you switched to debugging with the default builtin programming languages with default port number.

4. Watches

Watches used to evaluate variable or expression and their values can be realtime showing in stackview when debugging session paused, the nodes normally has a green colour and always placed on the top of stackview.

Caution:

a. Please ensure all source files have been dragged in the left side Treeview (Workspace Explorer) before start a debug session, because macOS app can't be allowed to access files outside of its sandbox.

b. When your source code file moved to another folder, you must drag the source code folder in Speare again then the debugging can correctly work.

C and C++ Debugger

The C and C++ debugger of Speare code editor implemented as a script client of [LLDB](http://lldb.lvm.org/) (<http://lldb.lvm.org/>), and supports extend it by yourself. The builtin module supports parsing modern C++ source code files including C++11 and C++14 syntax, e.g namespace, anonymous function, structure, union, class, enum etc and so many new features of the C++ programming language. You can enjoy debugging almost any type of C and C++ applications under the lightweight debugging environment of Speare code editor.

Start Debugging Steps:

1. Download Speare Debug Server:

→ http://sevenuc.com/download/c_debugger.tar.gz (10KB)

The source code of the C and C++ debugger can be view online here:

<https://github.com/chengdu/Speare> or here:

<https://sourceforge.net/projects/speare>

2. **Uncompress the tarball:**

Uncompress it to your local directory. e.g ~/Desktop and take a look at the readme.txt in it.

3. **Start the debug server:**

Please refer the readme.txt file.

4. **Debug session start:**

click "Start" button on the debug toolbar of Speare code editor.

Add breakpoint, step in, step out, step next, watch stack trace ...

5. **Run extra commands:**

Right click in the stackview (bottom left side) and then input any [lldb](#) command when the debugging session paused. Left click anywhere outside of the input box to close it and the command will be directly send to the debug server.

a. Add function breakpoints

. **breakpoint set --name functionname:** add a C function breakpoint.

. **breakpoint set --name classname::functionname:** add a C++ function breakpoint.

b. Process operation

- . `process attach --name xxx --waitfor`: attach another process by name.
- . `process attach --pid xxx`: attach another process by pid #xxx.

c. Thread operation

- . `thread list`: show all thread of current process.
- . `thread select 2`: select thread #2.
- . `thread backtrace all`: show thread info.
- . `register read`: read all CPU registers.
- . `thread step-inst`: step one machine instruction.
- . `thread step-over-inst`: step return one machine instruction.

d. Watchpoint operation

- . `watch list -v`: list watchpoints.
- . `watchpoint set variable x`: add a watchpoint x.

e. Frame operation

- . `frame list`: print all frame of the current thread.
- . `frame select 9`: select frame #9.

f. Display variable value

- . `frame variable x`: print x value.

...

Tips: Run to (run to meet a breakpoint), the source file that you want debugger stopped in it must already opened and has at least one breakpoint before run the command.

Modify the C and C++ debugger

You can directly modify the script client of lldb to satisfy your requirements.

Perl Debugger

The Perl debugger of Speare code editor implemented as a patched version of perl5db.pl, and support extend it by yourself. The debugger was based on the builtin debugger of Perl, so it can work with all versions of Perl interpreter that perl5db.pl supported.

Start Debugging Steps:

1. Download Speare Debug Server

→ http://sevenuc.com/download/perl_debugger.tar.gz (104KB)

The source code of the Perl debugger can be view online here:

<https://github.com/chengdu/Speare> or here:

<https://sourceforge.net/projects/speare>

2. Uncompress the tarball

Uncompress it to your local directory. e.g ~/Desktop and take a look at the readme.txt in it.

3. Start the debug server

```
$ cd ~/Desktop/debugger
```

```
$ perl -I ~/Desktop/debugger/Speare -d:Debugger fullpath.pl
```

```
* Warning: fullpath.pl the file must input with full path.
```

4. Debug session start

Click "Start" button on the debug toolbar of Speare code editor. Add breakpoint, step in, step out, step next, watch stack trace ...

5. Run extra commands:

Right click in the stackview (bottom left side) and then input any Perl debug command when the debugging session paused. Left click anywhere outside of the input box to close it and the command will be directly send to the debug server.

a. Add function breakpoints

. **b functionname:** add a function breakpoint.

b. Add condition breakpoints

. via breakpoint marker: Right click on the breakpoint, on the prompt menu, → select "Condition" and then input expression or use empty string to remove the condition, left click outside of the input box to close it and execute the command. e.g. `x > 5` means: Pause on the breakpoint only if `x > 5` is true.

. b fullpath.pl condition: e.g. `b /xxx/xxx/code.pl 6 $x > 5`.
`/xxx/xxx/code.pl`: fullpath of the script file. (do the same thing, optional)

c. Watchpoint operation

. w expr: add a watchpoint expr.
. W expr: delete watchpoint expr.
. W *: delete all watchpoints.

d. Evaluate express

. e expr: e.g. `"e $x+$y"`.

e. Display variable value

. p \$x: print value of variable x.
. p expr: print value of expression expr.

...

Switch Perl Interpreter

You can directly switch the Perl interpreter on the command line or debugging with your own self-compiled version of Perl.

Extend Speare Code Editor for Perl Development

Speare Code Editor can be easily extended to support any type of Perl development including web applications that based on web frameworks or any kinds of command line tool or standalone Perl applications.

To add scripts to better support Perl debugging in Speare code editor, please download the guide from here:
http://sevenuc.com/download/language_extension_protocol.pdf,
and following the description in it.