HOWTO Debug Lua extensions with GDB

The last time I've used an interactive debugger was in the school, Visual Basic 6.0 used to have a nice one. That was 8 years ago, then I went to scripting and web development so debugging changed dramatically for me. So in PHP is different process, it was a bit tricky to get used to “print\_r” debugging style instead of using breakpoints!

So, in the last few days, while struggling with [lua-cgic](http://gitorious.org/ophal/cgic) (my first ever Lua extension yet to be completed), I found an I/O bug that forced me to remember: the use of breakpoints, C programming and learn [GDB](http://www.gnu.org/software/gdb/). All those three in a row!

GDB is an interactive debugger, basically you load a program, set breakpoints, run it and wait for the program to stop at some breakpoint. That is obvious for some people, and awesome for others like me :), just have to clarify that everything happens into a console, if you miss the graphic interface, try the GDB GUI called [Insight](http://sourceware.org/insight/). <!–break–> OK, it is time to give some examples. Let's say you have compiled your Lua extension with the -g option, that means it has debugging information so that GDB can work. Now comes the tricky part:

Step 1: Load lua into GDB

$ gdb lua

GNU gdb (GDB) 7.4-debian

Copyright (C) 2012 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law. Type "show copying"

and "show warranty" for details.

This GDB was configured as "x86\_64-linux-gnu".

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>...

Reading symbols from /usr/bin/lua...(no debugging symbols found)...done.

(gdb)

There you see the GDB console! and now you can set some breakpoints:

Step 2: Set a breakpoint

(gdb) break myext\_foo

Function "myext\_foo" not defined.

Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 1 (myext\_foo) pending.

(gdb)

Of course GDB does not knows yet about “myext\_foo” function, but once you run your lua script, and it loads your extension, GDB will fire the breakpoint.

Step 3: Run your lua script

(gdb) run script.lua

Starting program: /usr/bin/lua script.lua

Breakpoint 1, myext\_foo () at myext.c:130

130 int myext\_foo() {

(gdb)

Awesome! Now you can debug your extension as any other C program.

Step 4: Debug

There are many tutorials out there:

* [A Quickstart Guide to Debugging C Programs with gdb](http://undergraduate.csse.uwa.edu.au/units/CITS2230/resources/gdb-intro.html)
* [How to Debug C Program using gdb in 6 Simple Steps](http://www.thegeekstuff.com/2010/03/debug-c-program-using-gdb/)

[Programming](http://www.devtome.com/doku.php?id=category:programming)