Tip 15: get `gdb` to print your structures

Ben Klemens

31 October 2011

level: intermediate debugger
purpose: look at the state of your data in different ways

GDB lets you define simple macros, which are especially useful for displaying nontrivial data structures—which is most of the work one does in a debugger. Gosh, even a simple 2-D array hurts your eyes when it's displayed as a long line of numbers.

The facility is pretty primitive. But you probably already wrote a C-side function that prints any complex structures you might have to deal with, so the macro can simply call that function with a keystroke or two. For example, I use `pd` to print `apop_data` structures via this macro:

```
define pd
    p apop_data_show($arg0)
end
document pd
Call apop_data_show to display an apop_data set.
E.g., for a data set declared with apop_data *d, use pd d.
end
```

Put these macros in your `.gdbinit`.

Notice how the documentation follows right after the function itself; view it via `help user-defined` or `help pd`. The macro itself just saves a few keystrokes, but because the primary activity in the debugger is looking at data, those little things add up.

To give a more involved example from the data structures I deal with in my work, the `apop_model` object has a list of settings groups—doesn't this already sound like a pain to inspect? So I wrote the following macro, with its accompanying documentation. This was enough to turn debugging these settings groups from pulling teeth to, um, cuddling puppies.

I don't expect you to follow the details of what it does, but as a somewhat exceptional case, you can see how much you can do: if argument one is `apop_mle`, then `$arg1_settings` → `apop_mle_settings`, and turning the same text into a string isn't an awkward exception like with C's macro processor.

```
define get_group
    set $group = ($arg1_settings *) apop_settings_get_grp( $arg0, "$arg1", 0 )
    p *$group
end
document get_group
Gets a settings group from a model.
Give the model name and the name of the group, like
get_group my_model apop_mle
and I will set a gdb variable named $group that points to that model, which you
like any other pointer. For example, print the contents with
p *$group
The contents of $group are printed to the screen as visible output to this macro
end
```

I partly needed this because you can't use preprocessor macros at the GDB prompt—they were subbed out long before the debugger saw any of your code, so if you have a valuable macro in your code, you may have to reimplement it in GDB. At least writing these things is quick.

For more examples, I put a list of my favorite gdb macros[1] for the GSL, SQLite, and Apophenia elsewhere on this site.

One last unrelated GDB tip, and then we can go back to C technique next time. Add this line to your `.gdbinit` to turn off those annoying notices about new threads:

```
set print thread-events off
```

[1]http://modelingwithdata.org/appendix_o.html#gdbinit