1

How to enable debugging for a plugin?

plugins debugging

I've created a plugin by following a tutorial from Gazebo website. Now I'd like to take it further but it is difficult to work without being able to set breakpoints and inspect variables. What steps do I have to do in order to be able to set breakpoints in my code?

asked Apr 4 '17
lubiluk
27 •5 •6 •8

add a comment

1 Answer

Sort by » oldest newest most voted

2

answered Apr 4 '17
Carlos Agüero
616 •3 •9

updated Apr 4 '17

I'll try to answer with an example:

Let's assume that you want to debug some code in the `LinearBatteryPlugin` that is shipped with Gazebo. You can launch Gazebo with the `linear_battery_demo.world` for loading the plugin. Let's also assume that your goal is to inspect the member variable `LinearBatteryPlugin::q` within the function `LinearBatteryPlugin::OnUpdateVoltage()`. Here are the steps for doing it using `gdb`:

- Make sure that you compiled Gazebo (or your plugin) with debug symbols. See this reference tutorial.
- Start `gdb` specifying `gzserver` as the executable program:

```
gdb gzserver
```

- Start `gzserver` with (assuming that your current directory is the top level directory of the Gazebo repository):

```
run worlds/linear_battery_demo.world
```

- You should see some `gdb` messages indicating that the program has started.
- Interrupt the execution of `gzserver` by pressing `CTRL-C`.
- We're going to create a breakpoint on a function. We can double check that `gdb` has the definition of the function (note that we have to indicate the entire namespace):

```
list gazebo::LinearBatteryPlugin::OnUpdateVoltage(const gazebo::common::BatteryPtr&)
```

- `gdb` should show a partial view of the source code:

```
 this->Init();
}

//////////////////////////////////////////////////
double LinearBatteryPlugin::OnUpdateVoltage(const common::BatteryPtr &_battery)
{
  double dt = this->world->GetPhysicsEngine()->GetMaxStepSize();
  double totalpower = 0.0;
  double k = dt / this->tau;

```

- Now, you can create your breakpoint:

```
break gazebo::LinearBatteryPlugin::OnUpdateVoltage(const gazebo::common::BatteryPtr&)
```

- `gdb` should confirm that the breakpoint was created. It's time to resume the `gzserver` execution:

```
continue
```

- Your execution should be interrupted when the code entered `OnUpdateVoltage()` and `gdb` should tell you that your breakpoint was hit:

```
Continuing.
[Switching to Thread 0x7fff797f7700 (LWP 20446)]

Thread 31 "gzserver" hit Breakpoint 1, gazebo::LinearBatteryPlugin::OnUpdateVoltage (this=
0x7fff44005a40,
_battery=
std::shared_ptr (count 4, weak 1) 0x180ed20) at
/home/caguero/workspace/gazebo/plugins/LinearBatteryPlugin.cc:128
128 {
```

- Now you can inspect your `q` variable:

```
print q
```

- And `gdb` should tell you the value:

```
(gdb) print q
$1 = 1.1660311540299606
```

Hopefully this is what you're looking for. Alternatively, if you're using Gazebo 8 we put in place an introspection system that is described in this tutorial (currently under review).

Comments

link

Thank you for your exhaustive answer. I could only add a clarification for future readers that if we want to debug only the plugin we're working on, we don't need to compile whole gazebo with debug symbols. Just a plugin is enough. We do this by using `cmake -DCMAKE_BUILD_TYPE=Debug ..` I have managed to set breakpoints in my plugin on standard, binary gazebo installation. Graphical debugging through an IDE also works well.

lubiluk (**Apr 7 '17**)

Hi lubiluk , could you please show how to set the breakpoint in a IDE? I managed to launch gzserver in both VSCode and Qt and compiled my plugin with Debug flag. But seems like the IDE doesn't associate the cpp file with the executable so the breakpoint I set in the code doesn't work. Thanks, Malcolm

Malcolm (**Apr 9 '17**)

Try using LD_PRELOAD. Assuming your sharedlib has debug symbols/etc, the source file absolute paths are encoded in the .debug_info section so gdb *will* find them. It may be that the sharedlibrary is not loaded until your URDF is loaded and the plugin is requested. You can use LD_PRELOAD environment variable to preload your libs (ex. libmyplugin.so) and include the path to your plugin in LD_LIBRARY_PATH. (Separate multiple files/paths by colons.)

guru-florida (**Oct 18 '0**)

Also ensure gazebo is not finding an alternative version of your plugin. For example, a colcon build version in an install folder, but you are building/debugging elsewhere. Confirm proper files/paths in gdb using:

```
(gdb) info sharedlibrary
(gdb) info sources
```

I did get debugging working for my plugin in gzserver after using LD_PRELOAD and this.

guru-florida (**Oct 18 '0**)

add a comment

Login/Signup to Answer