

Doug's Robotics Blog

Sunday, September 9, 2012

ROS, launch files, and parameters

I've been looking through some C++ code lately that uses a lot of parameters from roslaunch files. To make things clear, I decided to work through some examples.

Node Handles

When creating a ROS node, after calling `ros::init()`, the next step is to create a `ros::NodeHandle`. Both publishers and subscribers are created by objects returned from calls to the node handle.

The code I was looking at is part of a Phidgets stack (<https://launchpad.net/phidgets-ros-pkg>) and contains two different node handles in the same node:

```
ros::NodeHandle n;
ros::NodeHandle nh("~");
```

I had to ask myself why someone would create two node handles. And also, why the tilde on the second one? After some research, I discovered that the tilde refers to the node handle's namespace. This could be any namespace, which is roughly equivalent to a node name. However, the tilde refers to the current node's namespace, and is kind of like the "this" pointer in C++ or "self" in Python.

Why is this useful? In a roslaunch file, you can pass parameters to the system. There are "global" parameters that exist in the global, or default, namespace. These fall directly inside the `<launch>` tag. However, you can also put parameters under a `<node>` tag, and these will be in the local, or private, namespace of that node.

Ok, still clear as mud? Let's work through some explicit examples. I've put my example code up on GitHub so you are free to follow along. Go grab the files here : https://github.com/dougbot01/ros_param_test.

The Launch File

For simplicity, the same launch file is used in each case, except for the node denoted below by the "___".

```
<launch>
  <param name="serial" value="5" />

  <node name="___" pkg="ros_param_test" type="___" output="screen">
    <param name="serial" value="10" />
  </node>

</launch>
```

In the above file, you will see there are two parameters. The first parameter has the name "serial" with value 5. Since it is within the `<launch>` tag, this is in the global namespace. The second also has name "serial" with value 10, but it is in the `<node>` tag, and will go into the node's namespace. The point of having the same name is so we can see how to access values in different namespaces depending on the node handle.

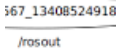
Case 1

Labels

AI **Arduino** [biped](#) [C++](#) [chassis](#) [compass](#)
[controls](#) [courses](#) [electronics](#) [firmware](#) [Fritzing](#) [gazebo](#)
[GitHub](#) [hardware](#) [IMU](#) [Kinect](#) [launch](#) [files](#) [line](#)
[follower](#) [load](#) [cell](#) [OpenNI](#) [parameter](#) [server](#) [Phidgets](#)
[Raspberry Pi](#) [robot](#) [news](#) **ROS** [rospy](#) [rosserial](#)
[RSSC](#) [search](#) [Skeleton_Tracking](#) [TurtleBot](#) [Ubuntu](#)
[walker](#) [Wixel](#) [Xtion](#) [ZBox](#) [Nano](#)

Popular Posts

ROS, launch files, and parameters

 Custom ROS messages with
 rosserial_arduino

ROS / Kinect Skeleton Tracking

Blog Archive

- [2014](#) (5)
- [2013](#) (5)
- ▼ [2012](#) (11)
 - ▼ [September](#) (3)
 - [ROS, launch files, and parameters](#)
 - [BYO Arduino, Less Frustrating Version](#)
 - [Biped Beginnings](#)
 - [July](#) (1)
 - [June](#) (1)
 - [April](#) (1)
 - [February](#) (3)
 - [January](#) (2)

About Me

dougbot01

[View my complete profile](#)

Followers

Followers (5)



[Follow](#)

Let's start with the basics and generate the node handle in the default global namespace. As shown in the code below, this is done by calling "ros::NodeHandle n;". After this line, there is a call 'n.getParam("serial", serial_number);' ... this gets the parameter "serial" and stores it in the variable "serial_number" which is initialized to -1 in the first line. The call to ROS_INFO() will print out the value obtained from the launch file.

file: src/param_test_global.cpp

```
#include <ros/ros.h>

int main(int argc, char* argv[])
{
    int serial_number = -1;
    ros::init(argc, argv, "parameter_test");

    //n is in the global namespace
    ros::NodeHandle n;
    n.getParam("serial", serial_number);

    ros::Rate loop_rate(10);
    ROS_INFO("Serial was %d", serial_number);
    while (ros::ok())
    {
        ros::spinOnce();
        loop_rate.sleep();
    }
}
```

Let's launch the node, with the following call ...

```
$ roslaunch ros_param_test case_1.launch
```

... and we would see the following output on the screen

```
process[param_test_global-2]: started with pid [11714]
[ INFO] [1347251105.803738523]: Serial was 5
```

As you can see, the global parameter was returned. Before killing the launch, open another terminal and type ...

```
$ rosparam dump dump.yaml
$ cat dump.yaml
```

This dumps all of the parameters to file "dump.yaml" and outputs them to screen. We see the following (with some output omitted):

```
param_test_global: {serial: 10}
serial: 5
```

Here we see how the parameters live in different namespaces.

Case 2

In this case, we'll change the node handle call to the node's namespace, as shown below with the tilde:

file: src/param_test_local.cpp

```
...
//n is in the node's namespace
ros::NodeHandle n("~");
n.getParam("serial", serial_number);
...
```

Let's run this code. Here you'll see we get the value 10 that lives in the node's namespace, declared in the <node> tag.

```
$ roslaunch ros_param_test case_2.launch
```

```
process[param_test_local-2]: started with pid [12023]  
[ INFO] [1347252255.817196129]: Serial was 10
```

Case 3

So, what if we want to use the node's namespace in our node handle, but access some parameters from the global namespace? ROS treats namespaces similar to a linux folder structure using the forward slash. The "root" is "/" and we prepend that to the parameter name to access a global parameter. This is "/serial" in line 4 below.

file: src/param_test_global_from_local.cpp

```
...  
//n is in the node's namespace  
ros::NodeHandle n("~");  
// get parameter from global namespace with "/"  
n.getParam("/serial", serial_number);  
...
```

```
$ roslaunch ros_param_test case_3.launch
```

```
process[param_test_global_from_local-2]: started with pid [12095]  
[ INFO] [1347252568.093727493]: Serial was 5
```

Case 4

And, what if we want to access a node variable from the global namespace? Using the linux folder structure analogy again, we need to go into the node's namespace and specify the parameter. This is declared as "/param_test_local_from_global/serial" in line 4 below.

file: src/param_test_local_from_global.cpp

```
...  
//n is in the global namespace  
ros::NodeHandle n;  
// get parameter from node's namespace  
n.getParam("/param_test_local_from_global/serial", serial_number);  
...
```

```
$ roslaunch ros_param_test case_4.launch
```

```
process[param_test_local_from_global-2]: started with pid [12162]  
[ INFO] [1347252635.929627917]: Serial was 10
```

Posted by [doughbot01](#) at 10:06 PM

 +1 Recommend this on Google

Labels: [launch files](#), [parameter server](#), [ROS](#)

1 comment:



Abhishek Goudar May 24, 2015 at 8:51 PM

Thanks for the post Doug! Just what I was looking for. The expound on namespaces was a bonus!

[Reply](#)