08. Roslaunch, ROS parameter szerver, Rosbag

Lecture

Roslaunch

- Launch multiple nodes
- Also launches ROS master if not running
- Set parameters
- XML file format, `.launch` extension

Example launch file

```
    <!-- dvrk_server.launch -->
    <!-- Launch the irob dVRK high-level robot controller. After start, it will wait for
irob_msgs/Robot actions -->

    <launch>
      <group ns="saf">

        <arg name="arm_typ" default="PSM2"/>
        <arg name="arm_name" default="arm_1"/>
        <arg name="camera_registration_file" default="registration_psm1.yaml"/>

        <arg name="instrument_info_file" default="prograsp_forceps.yaml"/>

        <include file="$(find irob_robot)/config/dvrk_topic_names.xml" />

        <node name="robot_server_$(arg arm_typ)" pkg="irob_robot"
type="robot_server_dvrk"
                                                output="screen">

          <param name="arm_typ" type="string" value="$(arg arm_typ)" />
          <param name="arm_name" type="string" value="$(arg arm_name)" />
          <param name="home_joint_angles" type="yaml" value="[0.0, 0.0, 0.0, 0.0, 0.0,
0.0]" />

          <rosparam command="load"
                file="$(find irob_robot)/config/$(arg camera_registration_file)"/>
```

```
        <rosparam command="load"
                file="$(find irob_robot)/config/$(arg instrument_info_file)"/>

    </node>
  </group>
</launch>
```

Usage

```
roslaunch package_name file.launch
roslaunch irob_robot dvrk_server.launch arm_typ:=PSM1
```

ROS Parameter Server

- Nodes can store and retrieve parameters at runtime
- Shared dictionary
- Best use for configuration
- ROS naming convention
- Private parameters (~)
- Available data types:
  - 32-bit integers
  - booleans
  - strings
  - doubles
  - iso8601 dates
  - lists
  - base64-encoded binary data
- Useful command: `rosparam`

Python API

```
# Call AFTER rospy.init_node()

# Getting parameters
global_name = rospy.get_param("/global_name")
relative_name = rospy.get_param("relative_name")
```

```
private_param = rospy.get_param('~private_name')
default_param = rospy.get_param('default_param', 'default_value')

# fetch a group (dictionary) of parameters
gains = rospy.get_param('gains')
p, i, d = gains['p'], gains['i'], gains['d']

# Setting parameters
# Using rospy and raw python objects
rospy.set_param('a_string', 'baz')
rospy.set_param('~private_int', 2)
rospy.set_param('list_of_floats', [1., 2., 3., 4.])
rospy.set_param('bool_True', True)
rospy.set_param('gains', {'p': 1, 'i': 2, 'd': 3})

# Using rosparam and yaml strings
rosparam.set_param('a_string', 'baz')
rosparam.set_param('~private_int', '2')
rosparam.set_param('list_of_floats', "[1., 2., 3., 4.]")
rosparam.set_param('bool_True', "true")
rosparam.set_param('gains', "{'p': 1, 'i': 2, 'd': 3}")

rospy.get_param('gains/p') #should return 1
```

Roslaunch API

```
<param name="arm_typ" type="string" value="ECM" />
<param name="publish_frequency" type="double" value="10.0" />
<rosparam command="load" file="FILENAME" />
```

YAML

- “A human friendly data serialization standard for all programming languages”

```
# registration_identity.yaml
t: [0.0, 0.0, 0.0]
R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
```

Rosbag

- Record and playback ROS topics
- Command line tool
- API for C++ and Python

```
rosbag record <topic_name>
rosbag record --all
rosbag play <filename.bag>
```

Practice

1: Marker: Disk

1. Create a new file named `dummy_cylinder.py` in the `scripts` folder. Publish disk shaped Marker with position (0.05, 0.05, -0.15) and radius of 0.1 m.

2: Launchfile and params for the markers

1. Create a new file named `dummy_markers.launch` in the folder `~catkin_ws/src/ros_course/launch`. If the folder does not exist, create that as well. Write a launchfile, that launches both dummy marker publisher nodes.

2. Modify the launchfile and the Python scripts so the dummy marker publishers receive the position of the marker as a ROS parameter, that can also be set from the command line (see the example in Chapter 6). Let the position of the markers have default values too, sphere: (-0.05, 0.1, -0.12), disk: (0.05, 0.05, -0.15).

3. Create a YAML file, containing the size and the color of the disk marker. Load those parameters in the Python script through roslaunch.

3: Navigation around the perimeter

1. Create a new launchfile named `psm_grasp.launch` for the script `psm_grasp.py`. Let dt, velocity and angular velocity of the jaws be set as ROS parameters.

2. Run `psm_grasp.launch` with different marker positions.

3. Modify `psm_grasp.py` so that the TCP moves around the perimeter of the disk marker before grasping the spherical one.

4: Record with rosbag

1. While running the program implemented in the previous exercise, record the contents of all topics to a bag file.

```
rosbag record --all
```

2. Install the package `rqt`.

```
sudo apt-get update
sudo apt-get install ros-noetic-rqt
sudo apt-get install ros-noetic-rqt-common-plugins
```

3. Play back the recorded bag file and echo some of the PSM1's topics (or visualize the coordinates of the PSM TCP using `rqt_plot`).

```
rosbag play <filename.bag>
rostopic echo /PSM1/measured_cp
```

Hasznos linkek

- Roslaunch

- ROS Parameter Server
- Python API for the ROS Parameter Server
- tag in roslaunch
- Rosparam YAML
- Rosbag
- rqt_plot