F1/10 Autonomous Racing

ROS - Launch and Bag files, + Turtlesim services

Lab Session 4 - CS4501/SYS4582 - Spring 2019

Instructor: Madhur Behl madhur.behl@virginia.edu
Course Website: https://linklab-uva.github.io/autonomousracing/
Git repo for this lab: https://github.com/linklab-uva/f1tenth-course-labs

---

Lab objectives:

In this lab, we will understand how to create and use `.launch` files; how to create and play `rosbags` , and how to invoke turtlesim services.

- Section [A]: Using `roslaunch`
- Section [B]: Recording and playing back data
- Section [C]: Using Services to draw letters with Turtlesim

Update your git repo first

The following instructions assume

- You created a `catkin_ws` folder on your machine, using the instructions discussed during the previous lab sessions.
- You created a `github` folder in your home directory and cloned the `f1tenth-course-labs` repository.
- You have sucessfully completed the Lab Session 4 with `beginner_tutorials`

Step 1)

Pull the latest code from the `f1tenth-course-labs` git repo:

```
cd ~/github
git pull
```

Notice that when you pull the new code, any additional files are linked with the `beginner_tutorials` package directory in you Catkin workspace folder.

[A] Using `roslaunch`

roslaunch starts nodes as defined in a launch file.

```
roslaunch [package] [filename.launch]
```

[A.1] >> `teleturtle.launch`

A launch file called `teleturtle.launch` is present in the `launch` sub-directory of the `beginner_tutorials` package:

```
 <launch>

  <group ns="turtlesim1">
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>

    <node pkg="turtlesim" name="teleop" type="turtle_teleop_key" launch-prefix=
"gnome-terminal -e"/>
  </group>

</launch>
```

Now let's roslaunch the launch file:

```
roslaunch beginner_tutorials teleturtle.launch
```

A turtlesim window will start along with a separate window for the `teleop_node` . rocore will also start on its own.

[A.2] >> `chat.launch`

Launch and examine the `chat.launch` file.

```
roslaunch beginner_tutorials chat.launch
```

[A.3] >> `turtlemimic.launch`

Lets examine another launch file: `turtlemimic.launch`

```
<launch>

  <group ns="turtlesim1">
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
    <node pkg="turtlesim" name="teleop" type="turtle_teleop_key" launch-prefix=
"gnome-terminal -e"/>
  </group>

  <group ns="turtlesim2">
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
  </group>

  <node pkg="turtlesim" name="mimic" type="mimic">
    <remap from="input" to="turtlesim1/turtle1"/>
    <remap from="output" to="turtlesim2/turtle1"/>
  </node>

</launch>
```

Two turtlesims will start and in a new terminal the teleop interface will show up. You you issue commands for one of the turtle to move, you will find that the second turtle will mimic those commands.

Now lets move the turtle in the `turtlesim1` instance using `rostopic pub`

```
rostopic pub /turtlesim1/turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.
0, 0.0]' '[0.0, 0.0, -1.8]'
```

You will see the two turtlesims start moving even though the publish command is only being sent to turtlesim1.

Notice how you didnt need to start `roscore` but it was started when you launched the launch file.

>> NOTE:

When we want a node to launch in a separate terminal window from the launch file we can use the `launch-prefix="gnome-terminal -e"` option in the `<node>` tag of the file.

[B] Recording and playing back data

Description: This part of the tutorial will teach you how to record data from a running ROS system into a `.bag` file, and then to play back the data to produce similar behavior in a running system.

[B.1] Recording data (creating a bag file)

Lets pull up the `turtlesim` ocean and the teleop node. First, execute the following commands in separate terminals:

Terminal 1:

```
roscore
```

Terminal 2:

```
rosrun turtlesim turtlesim_node
```

Terminal 1:

```
rosrun turtlesim turtle_teleop_key
```

OR use `roslaunch` to launch both the turtlesim nodes at the same time. </br>

[Q] Which launch file should we use ?

```
roslaunch beginner_tutorials XXXXXXX.launch
```

This will start two nodes - the turtlesim visualizer and a node that allows for the keyboard control of turtlesim using the arrows keys on the keyboard.

[B.2] Recording all published topics

Examine the full list of topics that are currently being published in the running system. To do this, open a new terminal and execute the command:

```
rostopic list -v
```

This should yield the following output:

```
Published topics:
 * /turtle1/color_sensor [turtlesim/Color] 1 publisher
 * /turtle1/cmd_vel [geometry_msgs/Twist] 1 publisher
 * /rosout [rosgraph_msgs/Log] 2 publishers
 * /rosout_agg [rosgraph_msgs/Log] 1 publisher
 * /turtle1/pose [turtlesim/Pose] 1 publisher

Subscribed topics:
 * /turtle1/cmd_vel [geometry_msgs/Twist] 1 subscriber
 * /rosout [rosgraph_msgs/Log] 1 subscriber
```

We now will record the published data. Open a new terminal window. In this window run the following commands:

```
mkdir ~/bagfiles
cd ~/bagfiles
rosbag record -a
```

Here we are just making a temporary directory to record data and then running rosbag record with the option -a, indicating that all published topics should be accumulated in a bag file.

Move back to the terminal window with turtle_teleop and move the turtle around for 10 or so seconds.

In the window running rosbag record exit with a Ctrl-C. Now examine the contents of the directory ~/bagfiles. You should see a file with a name that begins with the year, date, and time and the suffix .bag. This is the bag file that contains all topics published by any node in the time that rosbag record was running.

```
madhur@ubuntu:~/bagfiles$ ls
2019-02-12-12-18-13.bag
```

[B.3] Examining and playing the bag file

Now that we've recorded a bag file using rosbag record we can examine it and play it back using the commands rosbag info and rosbag play. First we are going to see what's recorded in the bag file. We can do the info command – this command checks the contents of the bag file without playing it back. Execute the following command from the bagfiles directory:

```
rosbag info <your bagfile>
```

You should see something like:

```
madhur@ubuntu:~/bagfiles$ rosbag info 2019-02-12-12-18-13.bag
path:        2019-02-12-12-18-13.bag
version:     2.0
duration:    16.1s
start:       Feb 12 2019 12:18:13.10 (1549991893.10)
end:         Feb 12 2019 12:18:29.22 (1549991909.22)
size:        150.4 KB
messages:    2007
compression: none [1/1 chunks]
types:       geometry_msgs/Twist [9f195f881246fdfa2798d1d3eebca84a]
             rosgraph_msgs/Log   [acffd30cd6b6de30f120938c17c593fb]
             turtlesim/Color     [353891e354491c51aabe32df673fb446]
             turtlesim/Pose      [863b248d5016ca62ea2e895ae5265cf9]
topics:      /rosout                               4 msgs    : rosgraph_msgs/Log
   (2 connections)
             /turtlesim1/turtle1/cmd_vel          19 msgs    : geometry_msgs/Twi
st
             /turtlesim1/turtle1/color_sensor   992 msgs    : turtlesim/Color
```

```
            /turtlesim1/turtle1/pose           992 msgs    : turtlesim/Pose
```

The next step in this tutorial is to replay the bag file to reproduce behavior in the running system. First kill the teleop program that may be still running from the previous section - a Ctrl-C in the terminal where you started turtle_teleop_key. Leave turtlesim running. In a terminal window run the following command in the directory where you took the original bag file:

```
rosbag play <your bagfile>
```

In this window you should immediately see something like:

```
madhur@ubuntu:~/bagfiles$ rosbag play 2019-02-12-12-18-13.bag
[ INFO] [1549996933.935808161]: Opening 2019-02-12-12-18-13.bag

Waiting 0.2 seconds after advertising topics... done.

Hit space to toggle paused, or 's' to step.
[DELAYED]  Bag Time: 1549991893.101184   Duration: 0.000000 / 16.123553   Delay

[RUNNING]  Bag Time: 1549991893.101184   Duration: 0.000000 / 16.123553
[RUNNING]  Bag Time: 1549991893.101184   Duration: 0.000000 / 16.123553
[RUNNING]  Bag Time: 1549991893.102541   Duration: 0.001357 / 16.123553
```

In its default mode rosbag play will wait for a certain period (.2 seconds) after advertising each message before it actually begins publishing the contents of the bag file. Waiting for some duration allows any subscriber of a message to be alerted that the message has been advertised and that messages may follow. If rosbag play publishes messages immediately upon advertising, subscribers may not receive the first several published messages. The waiting period can be specified with the -d option.

Eventually the topic /turtle1/cmd_vel will be published and the turtle should start moving in turtlesim in a pattern similar to the one you executed from the teleop program. The duration between running rosbag play and the turtle moving should be approximately equal to the time between the original rosbag record execution and issuing the commands from the keyboard in the beginning part of the tutorial. You can have rosbag play not start at the beginning of the bag file but instead start some duration past the beginning using the -s argument. A

final option that may be of interest is the -r option, which allows you to change the rate of publishing by a specified factor. If you execute:

```
rosbag play -r 2 <your bagfile>
```

You should see the turtle execute a slightly different trajectory - this is the trajectory that would have resulted had you issued your keyboard commands twice as fast.

[B.4] Recording a subset of the data

If any turtlesim nodes are running exit them and relaunch the keyboard teleop launch file:

```
rosrun turtlesim turtlesim_node
rosrun turtlesim turtle_teleop_key
```

In your bagfiles directory, run the following command:

```
rosbag record -O subset /turtle1/cmd_vel /turtle1/pose
```

The -O argument tells rosbag record to log to a file named subset.bag, and the topic arguments cause rosbag record to only subscribe to these two topics. Move the turtle around for several seconds using the keyboard arrow commands, and then Ctrl-C the rosbag record.

You may have noted that the turtle's path may not have exactly mapped to the original keyboard input - the rough shape should have been the same, but the turtle may not have exactly tracked the same path. The reason for this is that the path tracked by turtlesim is very sensitive to small changes in timing in the system, and rosbag is limited in its ability to exactly duplicate the behavior of a running system in terms of when messages are recorded and processed by rosrecord, and when messages are produced and processed when using rosplay. For nodes like turtlesim, where minor timing changes in when command messages are processed can subtly alter behavior, the user should not expect perfectly mimicked behavior.

[B.5] Combining bag files and launch files

In the `beginner_tutorials` package, there is a new sub-dirrectory available called `bags`

Examine the contents of this subdirectory using `rosls`

```
rosls beginner_tutorials/bags/
```

There is a bag file called `remapped_turtle.bag`

[Q] Use the rosbag program to inspect the file. What topics are included in the file?

Now write a new launch file called `turtle_remap.launch` in the launch directory of your package. This launch file will start the `turtlesim_node` and use the remap function to have that node listen to the topic in the `remapped_turtle.bag` file. The template for the launch file is shown below, but you will need to replace the "XXXXXX" with the appropriate topic:

```
<launch>
<node pkg="turtlesim" type="turtlesim_node" name="simulated_turtle">
    <remap from="turtle1/cmd_vel" to="XXXXXX" />
</node>
</launch>
```

After you fill in the correct value for XXXXXX in the launch file `turtle_remap.launch` , launch the file, and play the rosbag `remapped_turtle.bag`

[Q] What shape does the tutrle create in the ocean ?

Finally, close all running instances of the terminal and run the following launch file

```
roslaunch beginner_tutorials turtle_bag.launch
```

Everything will launch from a single file. Sit back and see the tutrle move according to the remapped cmd velocity in the bag file.

Examine, how we can play bag files from within the launch file.

[C] Using Services to draw letters with Turtlesim

In the `scripts` directory in the `beginner_tutorial` package. A new script called `turtleletterstwo.sh` is present that draws two letters in two different colors.

First make sure that the script is indeed executable.

- Navigate to the scripts sub directory using `roscd`
- Give the file executable privelages:

```
chmod +x turtleletterstwo.sh
```

Now run:

Terminal 1:

```
roslaunch beginner_tutorials teleturtle.launch
```

Terminal 2:

```
rosrun beginner_tutorials turtleletterstwo.sh
```

Understand what `turtleletterstwo.sh` is doing to draw thew letters.
This could be useful to asnwer some parts of the assignment, but you should write a python script rather than a shell script for the assignment.