Lab 1: Talker, Listener, and Messenger

The purpose of this lab is not make sure that you have access to your received laptop and it has been configured properly for ROS environments.

1 Change the `student` Password

The account that you will be using for the rest of this semester is `student`. The default password is `student`. After you login, open a terminal by pressing `Ctrl-Alt-t`. Then, change the password of the account `studnet` using the following command:

```
passwd student
```

and following the instruction on the screen. Your home directory is

```
/home/student
```

All your Python scripts should be at

```
/home/student/cs1567/src/mypackage/scripts
```

There are two ways to go to that directory. Manually using the `cd` command or type `roscd my` and press `Tab` and it will automatically be filled to

```
roscd mypackage/
```

Then type `scripts` and press Enter. Note that it should look like the following before you press Enter.

```
roscd mypackage/scripts
```

Note: `roscd` is a command that allowed you to change your working directory to any pre-installed packages or your package.

2 `talker.py`

Download `talker.py` from the CourseWeb which is located under "Labs" ⇒ "Lab 1:..." or use an editor of your choice to create a file named `talker.py` and type in the following:

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Save it and change the mode of the file to executable using the following command:

```
chmod +x talker.py
```

As explained in class, `talker.py` creates a node named `talker` and it periodically publish a message of type `String` onto the topic `chatter` every 0.1 second.

3 listener.py

Again, download the file `listener.py` or use an editor of your choice to create a file named `listener.py` and type in the following:

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber("chatter", String, callback)
    rospy.spint()

if __name__ == '__main__':
    listener()
```

Save it and change the mode of the file to executable using the following command:

```
chmod +x listener.py
```

As explained in class, `listener.py` creates a node named `listener` and it subscribes to the topic `chatter` which has type `String`. Every time it receives a message, it will show a message on the console screen.

4 Run Programs

1. Open two more terminals so that you have to total of three terminals.
2. On the first terminal, type the following command the run the ROS Master:

   ```
   roscore
   ```

   This is the ROS Master. It must be running when you want to execute any nodes.
3. On the second terminal, type the following command:

   ```
   rostopic list
   ```

   You should see the following on your console screen:

   ```
   /rosout
   /rosout_agg
   ```

   The command `rostopic list` lists all available topics published by all publisher nodes.
4. Again, on the second terminal, type the following command:

   ```
   rosrun mypackage talker.py
   ```

   The above command will execute the `talker` node. If your code has no error, your console screen should look like the following:

   ```
   [INFO] [WallTime: 1141544973.123456] hello world 114159973.12
   :
   [INFO] [WallTime: 1141544975.234567] hello world 114159975.23
   ```

   If you cannot see the similar message on the console screen, press `Ctrl-C` to shutdown the `talker` node. Then double check your source code and try to run it again.
5. Now, on the third terminal, list all topics using the command `rostopic list`. This time, you should see a new topic named `/chatter` as shown below.

   ```
   /chatter
   /rosout
   /rosout_agg
   ```

   If you do not see a new topic, there may be something wrong with your `talker.py`. Double check your source code. If you see the `/chatter` topic, type the following command:

   ```
   rostopic info chatter
   ```

   and you should see the following:

```
Type: std_msgs/String

Publishers:
 * /talker_2949_144154764130 (http://wall-e:54420/)

Subscribers: None
```

The command `rostopic info [topic]` is a tool for viewing information about a specific topic. One of the most important part is its type. Now, type the following command:

```
rostopic echo chatter
```

You should see the following messages on your console screen:

```
data: hello world 1441545678.12
---
data: hello world 1441545678.22
---
 :
```

The command `rostopic echo [topic]` is a very useful tool for monitoring messages published on a specific topic. Press `Ctrl-C` to stop.

6. Now, time to run the `listener` node. Type the following command:

```
rosrun mypackage listener.py
```

If all went well, you should see the following messages on your console screen:

```
[INFO] [WallTime: 1141548012.123456] /listener_3061_114154801012 I heard hello
world 1141548009.23
[INFO] [WallTime: 1141548013.123456] /listener_3061_114154801013 I heard hello
world 1141548010.23
  :
```

If you cannot see the above message, double check your `listener.py`.

5 Your Turn

1. Modify `talker.py` as follows:
   - Let `talker` node publishes messages on the topic `chatter1` instead of `chatter`.
   - This time, the message should simply be `"Hello from talker node"` without quotation marks (get rid of the time as well).
2. Modify `listener.py` as follows:
   - Let `listener` node subscribes to the topic `chatter2` instead of `chatter`.

- Whenever a message is receive, display the following on the console screen:

  ```
  Listener heard "[message]" from the messenger.
  ```

  where `[message]` is the message that it received from the topic `chatter2`

3. Create a `messenger` node (`messenger.py`) as follows:
   - This node should subscribes to the topic `chatter1`
   - This node should publishes to the topic `chatter2`
   - Whenever a message is received from `chatter1`, it should publish the following message

     ```
     Forwarding '[message]'
     ```

     where `[message]` is the message that it received from the topic `chatter1`

   The following is an example of the `messenger` node shown in class which can also be found in our CourseWeb. You can use this as a guideline for creating your `messenger` node:

   ```
   #!/usr/bin/env python

   import rospy
   from std_msgs.msg import String

   pub = rospy.Publisher('chatter2', String, queue_size=10)

   def messengerCallback(data):
       global pub
       messenger_str = "Messenger says %s" % data.data
       pub.publish(messenger_str)

   def messenger():
       rospy.init_node('messenger', anonymous=True)
       rospy.Subscriber('chatter', String, messengerCallback)
       rospy.spin()

   if __name__ == '__main__':
       messenger()
   ```

6 ssh to Given Laptop

The laptop that you received has been configured to be a `ssh` server. If you want to `ssh` to your given laptop, you must use CSSD's Virtual Private Network(VPN). The instruction can be found at `http://tech.cs.pitt.edu/faqs` under ”Software & Procedures” (item 3). This is very useful when you work as a team or try to create remote control robot. Note that your machine name will be `[robot's name].cs.pitt.edu` and according to the CSSD, it will only work when your given laptop is connected to the wireless router inside the robotic lab.