Robot Operating System
- Application Development

Maruthi S. Inukonda

18th Apr 2019

License

Agenda

- Workspace
- Build & Packaging
- Publisher and Subscriber nodes

Workspaces

Catkin

- A development and build framework for ROS applications
  - Greatly simplifies ROS application development.
  - Uses cmake, make
  - Ensures that custom messages comply to ROS standards.
  - Auto generated test utilities for unit testing.

Demo - Setting up workspace

- Create directory structure
  - Create a directory for workspace (eg. HADL)

```
$ mkdir -p ~/HADL/src
$ cd ~/HADL
$ catkin_make
```

Build & Packaging

Demo - Generating skeleton package

- Create catkin build/package files
  - Nodes could be developed in C++ and/or Python, setup for both.

```
$ cd ~/HADL/src
$ catkin_create_pkg myrospkg roscpp rospy
```

  - This automatically generates many files required for build & packaging.
  - Two important files are **`myrospkg/package.xml`** and **`myrospkg/CMakeLists.xml`**

Demo - Configuring package file

- Modify the auto-generated package file

```
$ vi ~/HADL/src/myrospkg/package.xml
```

- Change parameters in the package
  - Version, description, maintainer, license, etc

- If nodes use custom messages, add dependencies in the **`package.xml`**

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

Demo - Configuring cmake file

- Modify the auto-generated cmake file

  ```
  $ vi ~/HADL/src/myrospkg/CMakeLists.xml
  ```

- Add build dependencies in find_package(catkin) section of the **`CMakeLists.txt`**
  - If nodes use standard messages, add std_msgs in it.

    ```
    std_msgs
    ```
  - If nodes use custom messages, add message_generation in it.

    ```
    message_generation
    ```

- Add build targets under the “## Declare a C++ executable” section of the **`CMakeLists.txt`**

  ```
  add_executable(mypublisher src/mypublisher.cpp)
  target_link_libraries(mypublisher ${catkin_LIBRARIES})

  add_executable(mysubscriber src/mysubscriber.cpp)
  target_link_libraries(mysubscriber ${catkin_LIBRARIES})
  ```

Publisher Subscriber Nodes

Demo - Develop a publisher in C++ (1/2)

- Create a c++ file for publisher node

```
$ vi ~/HADL/src/myrospkg/src/mypublisher.cpp
#include <ros/ros.h>
#include <std_msgs/String.h>
```

- Register with ROS framework

```
ros::init(argc, argv, "mypublisher");
ROS_INFO("[ROS mypublisher] Node started.");
```

- Create a handle for communication & advertise the topic

```
ros::NodeHandle nh;
ros::Publisher pub = nh.advertise<std_msgs::String>("/mytopic", 1);
```

Demo - Develop a publisher in C++ (2/2)

- Keep publishing messages

```
ros::Rate loop_rate(1);
int count = 0;

while (nh.ok()) {
    std_msgs::String msg;
    std::stringstream ss;

    // publish the message
    ss << "hello world " << count++;
    msg.data = ss.str();

    pub.publish(msg);

    loop_rate.sleep();
}
```

http://wiki.ros.org/roscpp/Overview/Callbacks%20and%20Spinning

Demo - Develop a subscriber in C++ (1/2)

- Create a c++ file for subscriber node

```
$ vi ~/HADL/src/myrospkg/src/mysubscriber.cpp
#include <ros/ros.h>
#include <std_msgs/String.h>
```

- Define a call back function for each message type

```
void mysubscriberCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("Received: [%s].", msg->data.c_str());
}
```

- Register with ROS framework

```
ros::init(argc, argv, "mysubscriber");
ROS_INFO("[ROS mysubscriber] Node started.");
```

Demo - Develop a subscriber in C++ (2/2)

- Create a handle for communication. Subscribe to a topic with a queue size, and the callback function.

```
ros::NodeHandle nh;
ros::Subscriber sub = nh.subscribe("/mytopic", 1000,
                                   mysubscriberCallback);
```

- Keep looping in framework, until node is killed.

```
ros::spin();
```

This model of using framework is called Inversion of control.

Demo - Build & Run nodes

- Build the nodes

```
$ cd ~/HADL
$ catkin_make
```

- Run the nodes
  - Start the subscriber

```
$ ~/HADL/devel/lib/myrospkg/mysubscriber
```

  - List nodes, topics

```
$ rosnode list
$ rostopic list
```

  - Start the publisher

```
$ ~/HADL/devel/lib/myrospkg/mypublisher
```

References

References

- ROS Tutorials
- ROS Publisher/Subscriber in C++
- ROS Publisher/Subscriber in Python

Q & A