ROS

ITMO more than a UNIVERSITY

Операционная система ROS

Разработка программных решений для роботов

План занятия

- Опрос на15 мин
- Создание ROS пакетов
  - Зависимости + Rosdep
  - Настройка пакета
  - Инструменты логирования
  - Файлы запуска
  - Имена и пространства имен
  - Сервер параметров
  - Создание пользовательских типов сообщений
- 10 мин на вопросы по лабораторным

Создание пакета

Существует 2 инструмента для создания пакетов в ROS: roscreate-pkg и catkin , далее мы будем рассматривать второй вариант.

К пакету предъявляются всего 3 требования:

- Каждый пакет должен располагаться в отдельной папке
- Пакет должен содержать файл package.xml, совместимый с catkin
- Пакет должен содержать файл CMakeLists.txt, который использует catkin

Для создания пакета можно воспользоваться командой catkin_create_pkg
catkin_create_pkg my_robot_super_controller dependency_package_name

- my_robot_super_controller - название пакета
- После названия пакета можно указать сразу все зависимости, и соответствующие записи появятся в служебных файлах пакета.

Файл package.xml содержит метаинформацию о пакете.
Если это метапакет, то CMakeLists.txt должен иметь соответствующий шаблонный файл CMakeLists.txt.

Зависимости

Чаще всего используемыми зависимостями являются std_msgs rospy roscpp

Зависимости, которые мы указываем при создании пакета являются зависимостями первого порядка, их можно просмотреть с помощью инструмента rospack.

rospack depends1 my_robot_super_controller

В большинстве случаев зависимость также будет иметь свои собственные зависимости (**косвенные зависимости**). Например, у rospy есть и другие зависимости.

rospack depends1 rospy

Пакет может иметь довольно много косвенных зависимостей. Но rospack может рекурсивно определять все вложенные зависимости.

rospack depends my_robot_super_controller

Необычный факт: пакет rospy имеет зависимость от пакета roscpp

Ссылка на официальную документацию: http://wiki.ros.org/ROS/Tutorials/CreatingPackage

Rosdep

Rosdep является автономным инструментом командной строки для установки системных зависимостей, который вы можете загрузить и использовать отдельно от ROS.
Установка: sudo apt-get install python3-rosdep
Инициализация (один раз): sudo rosdep init
Обновление: rosdep update (!!! Без sudo!!!)

Установка зависимостей отдельного пакета:

```
rosdep install AMAZING_PACKAGE
```

Установка зависимостей всех пакетов (в корневой папке пакета):

```
rosdep install --from-paths src --ignore-src -r –y
```

Ссылка на вики: https://docs.ros.org/en/independent/api/rosdep/html/commands.html

Настройка пакета

Манифест пакета представляет собой XML-файл с именем **package.xml**, который должен быть включен в корневую папку любого пакета, совместимого с Catkin (создается автоматически при использовании catkin). Этот файл определяет свойства пакета, такие как имя пакета, номера версий, авторов, сопровождающих и зависимости от других пакетов Catkin.

Структура файла:

```
<package format="1">
</package>
```

Формат: Существует 1 и 2 формат, в зависимости от используемой версии необходимо использовать специфичные для версии теги. Формат 2 является рекомендованным, однако это не ограничивает использование формата 1.

Основные различия представлены в данном документе
https://docs.ros.org/en/melodic/api/catkin/html/howto/format2/migrating_from_format_1.html

Теги в **package.xml**

Минимальный набор тегов (формат 1):
- <name> - Название пакета
- <version> - Номер версии пакета (должно быть целым числом, разделенным тремя точками), например `1.2.3`
- <description> - Описание содержимого пакета
- <maintainer> - Имя человека(ов), который/обслуживает пакет
- <license> - Лицензия(и) на программное обеспечение (например, GPL, BSD, ASL), под которой выпущен код.

Дополнительные теги

• <url> - Ссылка на сайт с информацией о пакете
• <author> - Имена авторов

Теги зависимостей:
<buildtool_depend>catkin</buildtool_depend>
<build_depend>rospy</build_depend>
<exec_depend>rospy</exec_depend>

Ссылка на вики: http://wiki.ros.org/catkin/package.xml

CMakeLists.txt

Файл CMakeLists.txt является входными данными для системы сборки CMake для создания пакетов программного обеспечения.
Любой пакет, совместимый с CMake, содержит один или несколько файлов CMakeLists.txt, в которых описывается, как собрать исходный код и куда его установить.

Файл CMakeLists.txt, используемый для проекта catkin, представляет собой стандартный файл CMakeLists.txt с несколькими дополнительными ограничениями.

_____

CMake — это кроссплатформенное семейство инструментов, предназначенное для создания, тестирования и упаковки программного обеспечения. CMake используется для управления процессом компиляции программного обеспечения с использованием простых файлов конфигурации.

- cmake_minimum_required(VERSION 2.8.3) (Catkin поддерживает версию 2.8.3 или выше.)
- project(robot_brain) - Название пакета передается в функцию project (на это название можно ссылать используя ${PROJECT_NAME} далее в файле)
- find_package(catkin REQUIRED)` - прочие CMake пакеты

Инструменты логирования

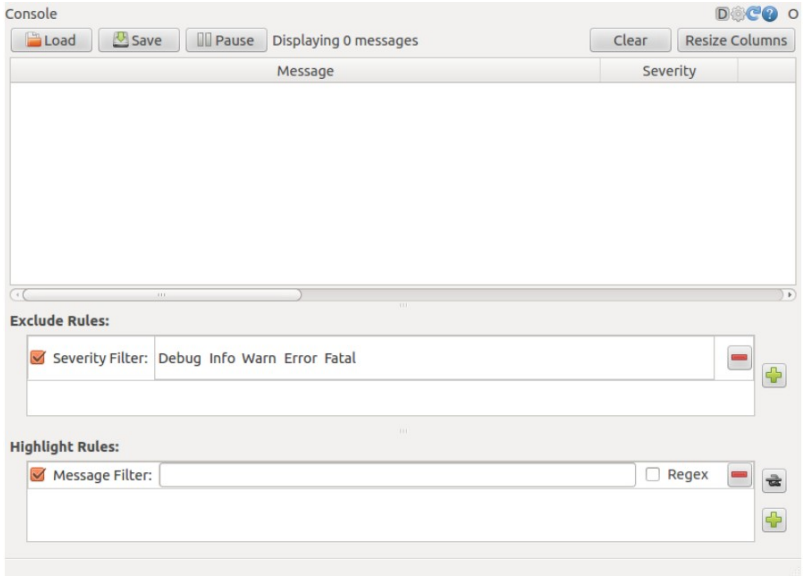
rqt_console & rqt_logger_level

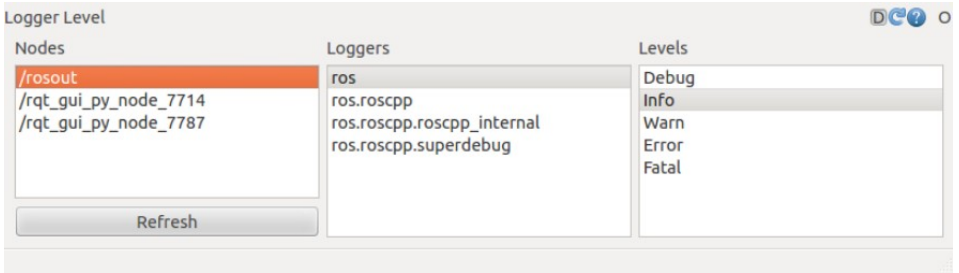
Чтобы изменить уровень логирования необходимо поменять Level в окне Logger Level и нажать Refresh.

Каждый уровень логирования имеет приоритет:

- Fatal (самый высокий)
- Error
- Warn
- Info
- Debug (самый низкий)

Вспомним про похожий немного
rqt_graph и rqt_plot





Файлы запуска **roslaunch**

Файлы типа .launch предназначены для одновременного запуска нескольких узлов (процессов) и располагаются в папке **launch**

Рассмотрим на примере:
<launch> - тип разметки (сопровождается закрывающим тегом </launch**>)**
<group ns="turtlesim1"> - группа ресурсов (с указаниеим пространства имен)
<node pkg="turtlesim" name="sim" type="turtlesim_node"/> - запуск ноды из пакета (одной строкой).
<node pkg="turtlesim" name="mimic" type="mimic"> - второй способ запуска ноды из пакета (в две строки)
<remap from="input" to="turtlesim1/turtle1"/> - вложенный тег переадресации (внутри ноды)
</node>- закрывающий тэг запуска ноды

```
<launch>
    <!--<node pkg="turtlesim" name="sim" type="turtlesim_node"/>-->
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>

</launch>
```

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

Пример запуска файла
roslaunch package_name robot_setup.launch

```
<launch>
    <node name="listener" output="screen"/>
</launch>
```

Имена и пространства имен

Имена ресурсов графа — это важный механизм в ROS, обеспечивающий **инкапсуляцию**. **Каждый ресурс определяется в пространстве имен**, которое он может использовать совместно со многими другими ресурсами.

Соединения могут устанавливаться между ресурсами в разных пространствах имен, но обычно это делается с помощью кода интеграции над обоими пространствами имен. Эта инкапсуляция изолирует различные части системы от случайного захвата ресурса с неверным именем или глобального перехвата имен.

К названиям ресурсов предъявляются следующие требования:

- Первый символ – это буква ([a-z|A-Z]), тильда (~) или косая черта (/).
- Последующие символы могут быть буквенно-цифровыми ([0-9|a-z|A-Z]), подчеркиванием (_) или косой чертой (/).
- Исключение: базовые имена (описанные ниже) не могут содержать косую черту (/) или тильды (~).

В ROS существует четыре типа имен ресурсов графа: базовые, относительные, глобальные и частные, которые имеют следующий синтаксис:

- base
- relative/name
- /global/name
- ~private/name

Имена, начинающиеся с «/», являются глобальными. Глобальных имен следует избегать, насколько это возможно, поскольку они ограничивают переносимость кода.

Ссылка на вики: http://wiki.ros.org/Names

Remapping

Любое имя внутри узла ROS можно переназначить при запуске узла.

Пример: <remap from="/different_topic" to="/needed_topic"/>

Атрибуты:

- from="original-name"
- to="new-name"

Любое имя ROS внутри узла **можно переназначить при его запуске из командной строки**. Это функция позволяет запускать один и тот же узел в нескольких конфигурациях из командной строки. Все имена ресурсов можно переназначить. Эта функция ROS позволяет отложить назначение сложных имен до фактической загрузки системы во время выполнения.

Пример переадресации при запуске:
rosrun rospy_tutorials talker chatter:=/wg/chatter

Ссылка на вики: http://wiki.ros.org/roslaunch/XML/remap

Сервер параметров

Сервер параметров — это общий многовариантный словарь, доступный через сетевые API. Узлы используют этот сервер для хранения и получения параметров во время выполнения.
Его **лучше использовать для** статических, недвоичных данных, **таких как параметры конфигурации**. Он предназначен для глобального использования, чтобы инструменты могли легко проверять состояние конфигурации системы и при необходимости изменять ее.

Сервер параметров реализован с использованием XMLRPC и работает внутри ROS Master, его API доступен через обычные библиотеки XMLRPC.

Получение параметров:

```
global_name = rospy.get_param("/global_name")
```

Установка параметров:

```
rospy.set_param('a_string', 'baz')
```

Типы данных XMLRPC:

- 32-bit integers
- booleans
- strings
- doubles
- iso8601 dates
- lists
- base64-encoded binary data

Ссылка на вики: http://wiki.ros.org/Parameter%20Server
http://wiki.ros.org/rospy/Overview/Parameter%20Server

Типы сообщений

.msg - представляют собой простые текстовые файлы, описывающие поля сообщения ROS (тип поля и имя поля в каждой строке). Они используются для генерации исходного кода сообщений на разных языках.
Сами файлы msg хранятся в каталоге пакета, в подпапке msg.

Типы полей, которые вы можете использовать (примитивы):

- int8, int16, int32, int64 (plus uint*)
- float32, float64
- string
- time, duration
- other msg files
- variable-length array[] and fixed-length array[C]

Создание нового типа сообщений

- **Что понадобится?**

  - Умение создавать папки
  - Умение создавать текстовый файл
  - Подключить message_generation + message_runtime в **package.xml**
  - Добавить зависимости в CMakeLists.txt

В рамках подготовки к ЛР 2 нам необходимо будет создать пользовательский тип сообщений, новый пакет и новую ноду паблишер + сабскрайбер.

Пример создания файла Num.msg

- mkdir msg
- $ echo "int64 num" > msg/Num.msg

После генерации необходимых файлов, появится новый тип сообщений Num

ITMO re than a UNIVERSITY

Спасибо за внимание!