

ROS

Основные понятия

Клонируем проект с git

- Создаем директорию - рабочую директорию (workspace)
 - `mkdir <имя_рабочей_дирректории>`
 - `cd <имя_рабочей_дирректории>`
- Клонировем репозиторий в поддиректорию `src`
 - `git clone https://github.com/AndreyMinin/ros_course.git src`
- Обновляем проект
 - `git pull`

Сборка ROS проекта

- Переходим в рабочую директорию
 - `cd <имя_рабочей_области>`
- Создание своего проекта
 - `cd src`
 - `catkin_create_pkg <имя проекта> <зависимости e.g. : ros_cpp std_msgs>`
- Сборка проекта
 - `cd <имя_рабочей_области>`
 - `catkin_make`

Запуск модулей ROS

- `roscore`
- Инициализация workspace
 - `cd <workspace>`
 - `$source devel/setup.bash`
- `roslaunch chat listener`
- `roslaunch chat talker`

Имена модулей

- `ros::init(argc, argv, "talker");`
- Имена должны быть уникальными
- При повторном запуске модуля с существующим именем - первый модуль завершается
- Изменить имя можно с помощью параметров запуска
 - `roslaunch chat talker __name:=talker1`
 - `roslaunch chat talker __name:=talker2`

Имена топиков

- Имена топиков издателя и подписчика должны совпадать
- Типы топиков также должны совпадать
- Имена часто топиков подчиняются иерархии
- Изменение наименования топиков - remap
 - `roslaunch chat talker chatter:=/robot/mychat`

roslaunch - файл запуска

- roslaunch - средство запуска нескольких узлов с указанием их параметров
- roslaunch оперирует специальными имеющими формат xml файлами (.launch), в которых указываются какие узлы с какими параметрами запускать
- roslaunch имя_пакета имя_файла.launch
- roslaunch автоматически запускает roscore

launch файл для talker и listener

```
<launch>
  <node name="mytalker" pkg="chat" type="talker_node" output="screen" />
  <node name="mylistener" pkg="chat" type="listener_node" output="screen"
/>
</launch>
```

```
<!--
```

name - имя узла , может быть произвольным

pkg - имя пакета (catkin_create_pkg)

type - имя исполняемого файла (add_executable(имя ...))

output - управление выводом в консоль (screen или log) -->

launch файл для talker и listener

```
<launch>
  <node name="mytalker" pkg="chat" type="talker" output="screen">
    <remap from="chat" to="mychat" />
  </node>
  <node name="mylistener" pkg="chat" type="listener" output="screen" />
</launch>
```

Отладочный вывод

- Специальные макросы (printf)
 - `ROS_DEBUG("debug :%d", int_var)`
 - `ROS_INFO`
 - `ROS_ERROR`
 - `ROS_WARN`
- Макросы для работы с потоком вывода (`std::cout`)
 - `ROS_xxx_STREAM("debug " << int_var)`
- Вывод с указанием времени
- Вывод в консоль, в файл лога и в специальный топик `ros_out`
- Управление уровнем логгирования
 - Через `rqt`
 - С помощью конфигурационного файла

Задание конфигурации логирования

```
<launch>  
  <env name="ROSCONSOLE_CONFIG_FILE" value="$(find chat)/cfg/rosconsole.conf"  
/>  
  
  <node name="mytalker" pkg="chat" type="talker" output="screen" />  
  <node name="mylistener" pkg="chat" type="listener" output="screen" />  
</launch>
```

Задание конфигурации логирования

Содержимое файла chat/cfg/rosconsole.conf

```
log4j.logger.ros=INFO
```

```
log4j.logger.ros.chat=DEBUG
```

Модель робота в среде stage

- apt-get install ros-indigo-stage-ros
- Модель двухколесного робота (дифференциальный привод)
- Файлы запуска и конфигурации модели в проекте stage_launch
- Запуск модели из рабочей папки
 - roslaunch stage_launch task1.launch

Топики модели

/cmd_vel - управление скоростью робота

Тип топика geometry_msgs/Twist - сообщение содержит трехмерные заданные линейную и угловую скорости

geometry_msgs/Vector3 linear

float64 x

float64 y

float64 z

geometry_msgs/Vector3 angular

float64 x

float64 y

float64 z

Топики модели

`odom(base_pose_ground_truth)` - текущая позиция робота по одометрии, тип `geometry_msgs/Odometry`

`geometry_msgs/TwistWithCovariance`

`geometry_msgs/Twist` - текущая скорость

`float64[36] covariance` - параметры точности

`geometry_msgs/PoseWithCovariance`

`geometry_msgs/Pose`

`geometry_msgs/Point (x y z)` - положение робота

`geometry_msgs/Quaternion(x, y, z, w)` - ориентация $2 \cdot \text{atan2}(z, w)$

`float64[36] covariance` - параметры точности

Топики модели

base_scan - данные дальномера

Тип sensor_msgs/LaserScan

float32[] ranges - массив (вектор) дальностей

ranges.size() - количество измерений в скане

ranges[i] - доступ к измерению (дальность i - го луча)

float32 angle_min - минимальный угол (начало скана относительно оси x)

float32 angle_max - максимальный угол (конечный угол сканирования)

float32 angle_increment - угловое разрешение

Текст программы