# ROS

Основные понятия

# Клонируем проект с git

- Создаем директорию - рабочую директорию (workspace)
  - mkdir <имя_рабочей_дирректории>
  - cd <имя_рабочей_дирректории>
- Клонируем репозиторий в поддирректорию 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
- rosrun chat listener
- rosrun 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
  - rosrun chat talker __name:=talker1
  - rosrun 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
  - rosrun 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 каким 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
```