ROS
Введение

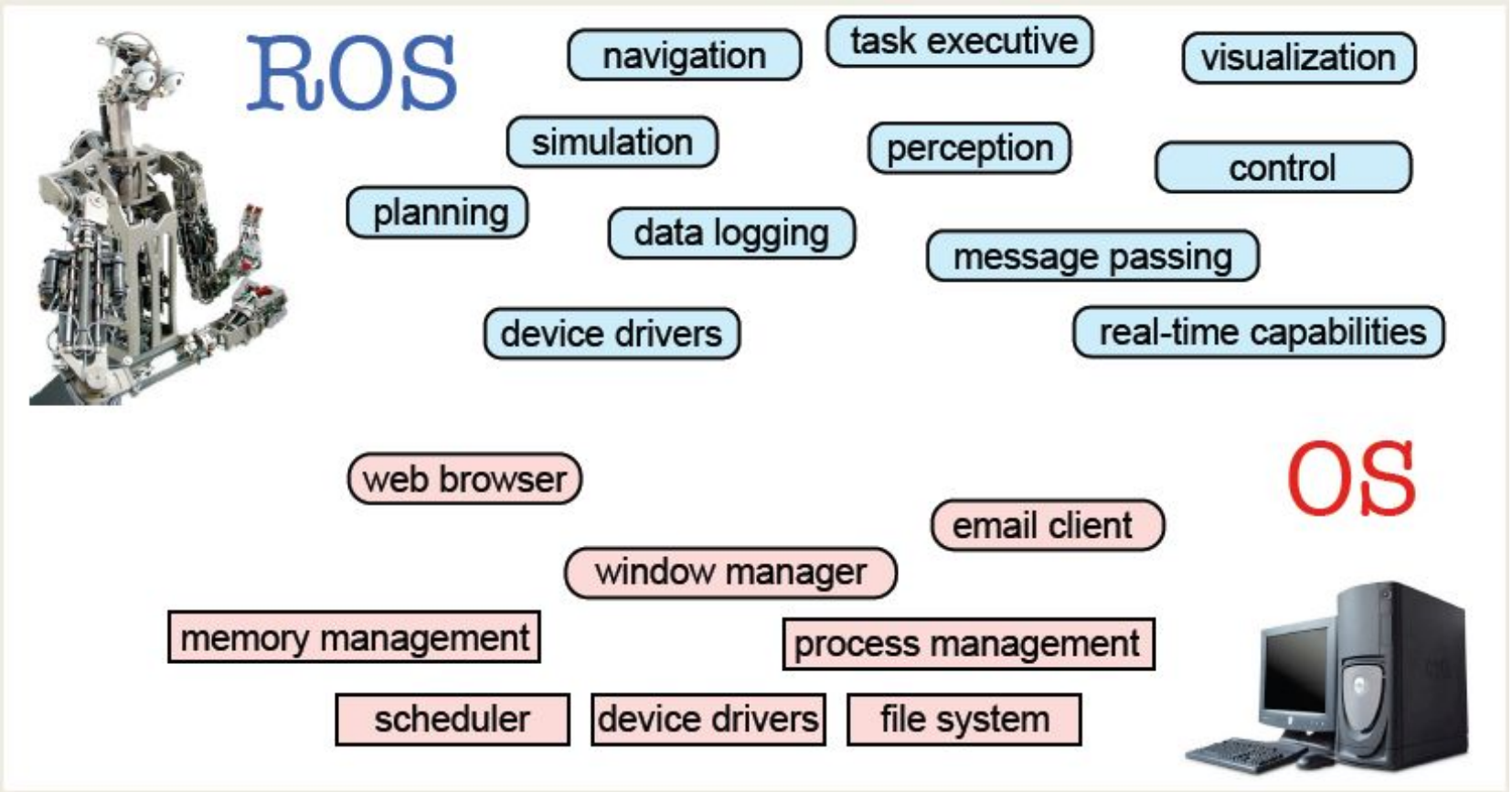
Что такое ROS ?

- Robot Operating System
- Коллекция библиотек (управления мр и манипуляторами, работа с стз)
- Инфраструктура обеспечивающая запуск и работу, отладку распределенных систем
- Система разработки и управления пакетами
- Коллекция драйверов для различных устройств (камеры, дальномеры ...)
- Сообщество разработчиков (www.ros.org-wiki, tutorials)

Основные возможности

- ROS как операционная система
  - hardware abstraction
  - sensor drivers
  - межпроцессный обмен
  - системные утилиты
- ROS как система разработки ПО
  - библиотеки, реализующие основной функционал и алгоритмы ПО для роботов (SLAM, planning, vision)
  - система управления пакетами (установка, сборка, зависимости)

ROS - надстройка над ОС



Taken from Sachin Chitta and Radu Rusu (Willow Garage)

Идеология ROS

- Peer to peer: система строится как состоящая из отдельных равноправных модулей (обмен сообщениями)
- Простые утилиты. Отдельные небольшие программы для визуализации, диагностики системы, логирования.
- Поддержка нескольких языков разработки: C++, Python, Java(Android), Matlab
- Free and open source

Основные концепции ядра ROS

- TCP/UDP
- nodes
- topic
- messages
- services
- parameters
- packets

Nodes

- node (узел ROS) - пользовательская программа, решающая определенную задачу (взаимодействие с устройством, обработка данных, построение карты и т.п.)
- могут публиковать сообщения и подписываться на сообщения других узлов
- взаимодействуют с ROS с помощью библиотек roscpp (ros c++ client library) или с помощью rospy (python)

ROS Topic

- Topic - имя для потока сообщений определенного типа:
  - например сообщения от узла-драйвера камеры с именем /camera/image с типом CompressedImage (сжатое изображение)
- Узлы ROS могут публиковать сообщения и подписываться на сообщения других узлов
- 1:N publish/subscribe: в один топик можно писать читать из разных узлов

ROS Message

- типизированная структура данных для обмена между узлами ROS
- Например geometry_msgs/Twist
  - Vector3 linear
  - Vector3 angular
- Стандартные сообщения (std_msgs, geometry_msgs, sensor_msgs, nav_msgs ...)
  - http://wiki.ros.org/std_msgs
  - ...
- Пользовательские сообщения

ROS Service

- взаимодействие client - server
- запрос и ответ

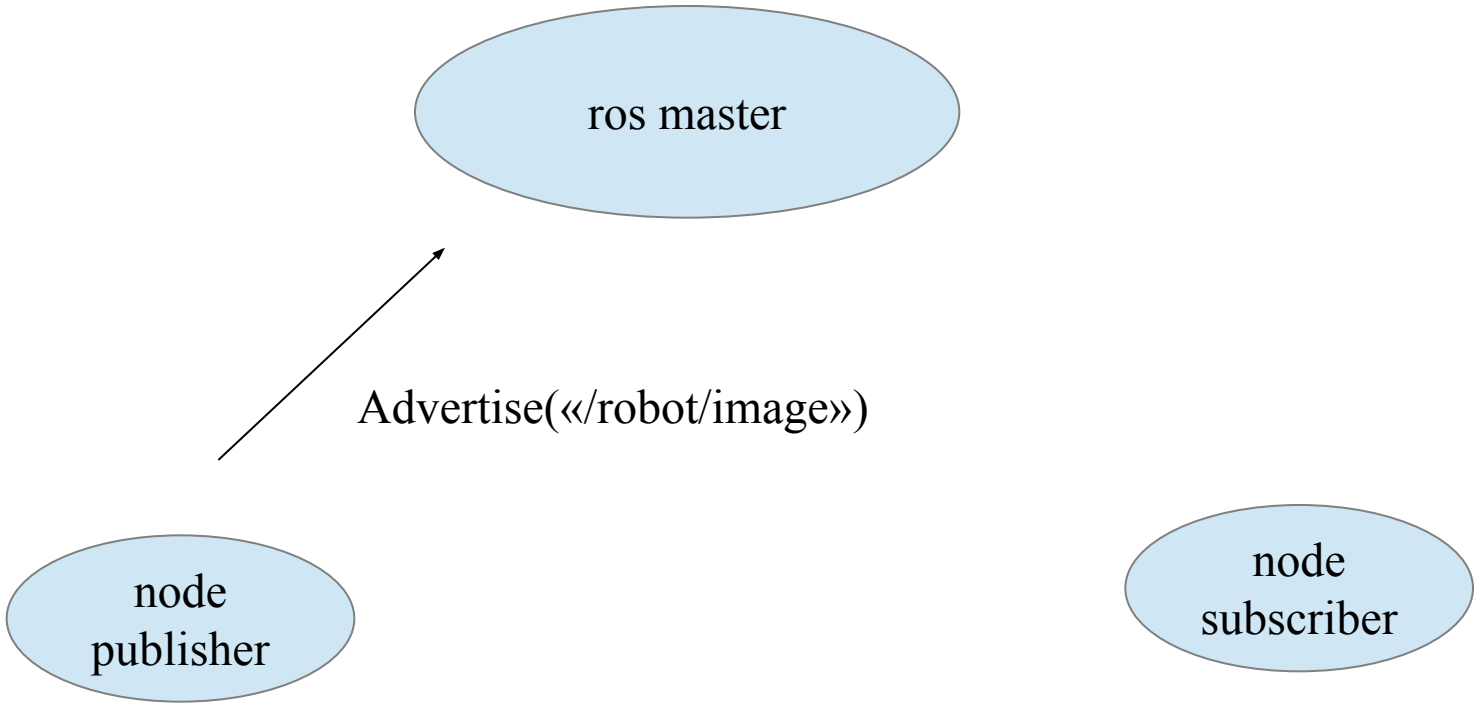
ROS CORE

- ros master
- ros parameter service
- ros out

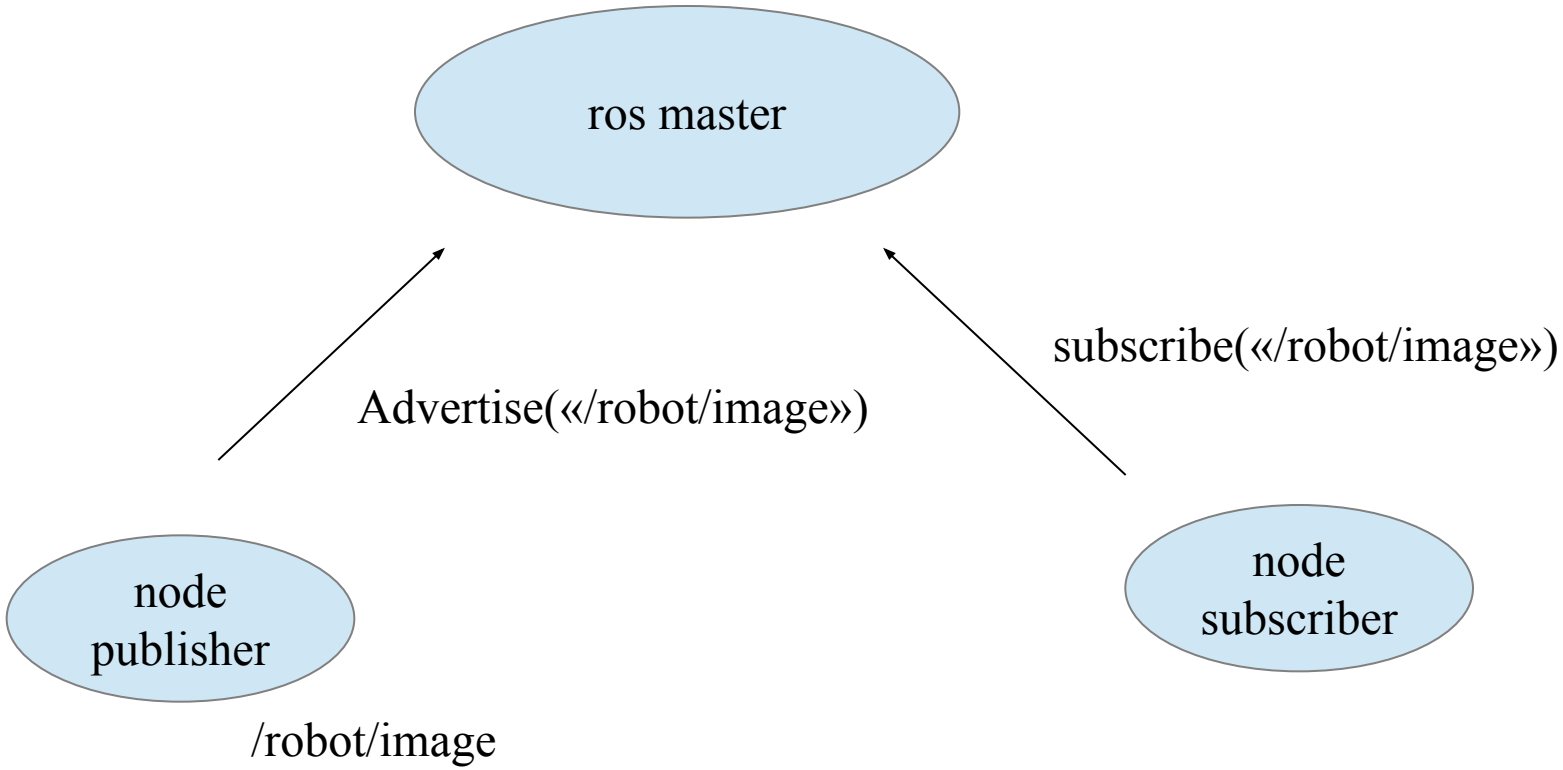
Роль ROS Master

- Обеспечивает узлы информацией необходимой для отправки - получения сообщений
  - Каждый узел при старте соединяется с master для регистрации топиков сообщений, которые узел будет публиковать и на которые он хочет подписаться

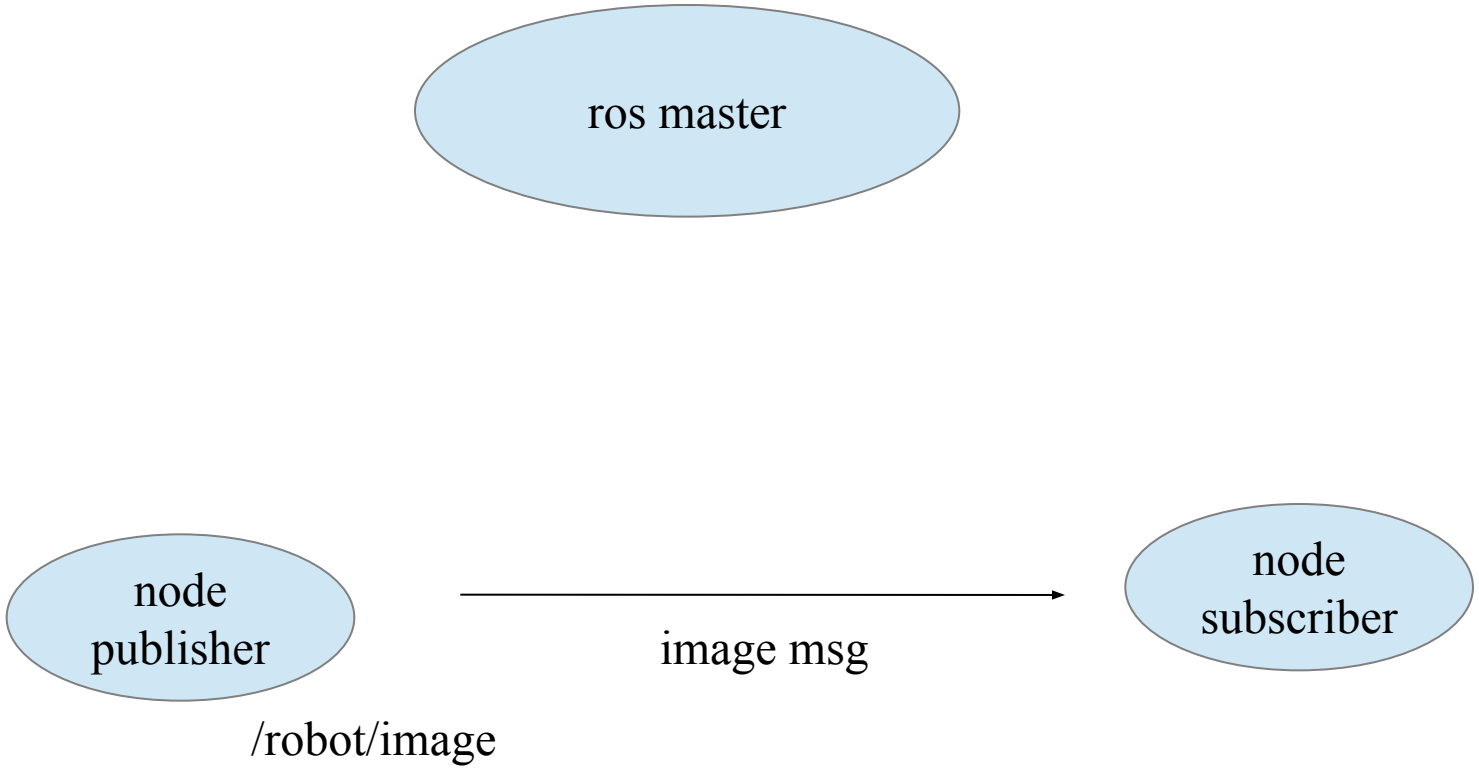
Роль ros-master



Роль ros-master



Роль ros-master



Сервер параметров

- Доступная всем узлам сети ROS база данных (ключ-значение)
- Используется для задания конфигурационных параметров
- локальные и глобальные параметры

Запуск процессов nodes

- rosrun
- roslaunch

Средства диагностики

- rostopic list, echo, hz, pub
- rosnode list
- rosservice list
- rqt
- rviz

Subscriber

```
#include "ros/ros.h"
#include "std_msgs/String.h"

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

int main(int argc, char **argv){
   ros::init(argc, argv, "listener");

   ros::NodeHandle n;

  ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);
  ros::spin();
  return 0;
}
```

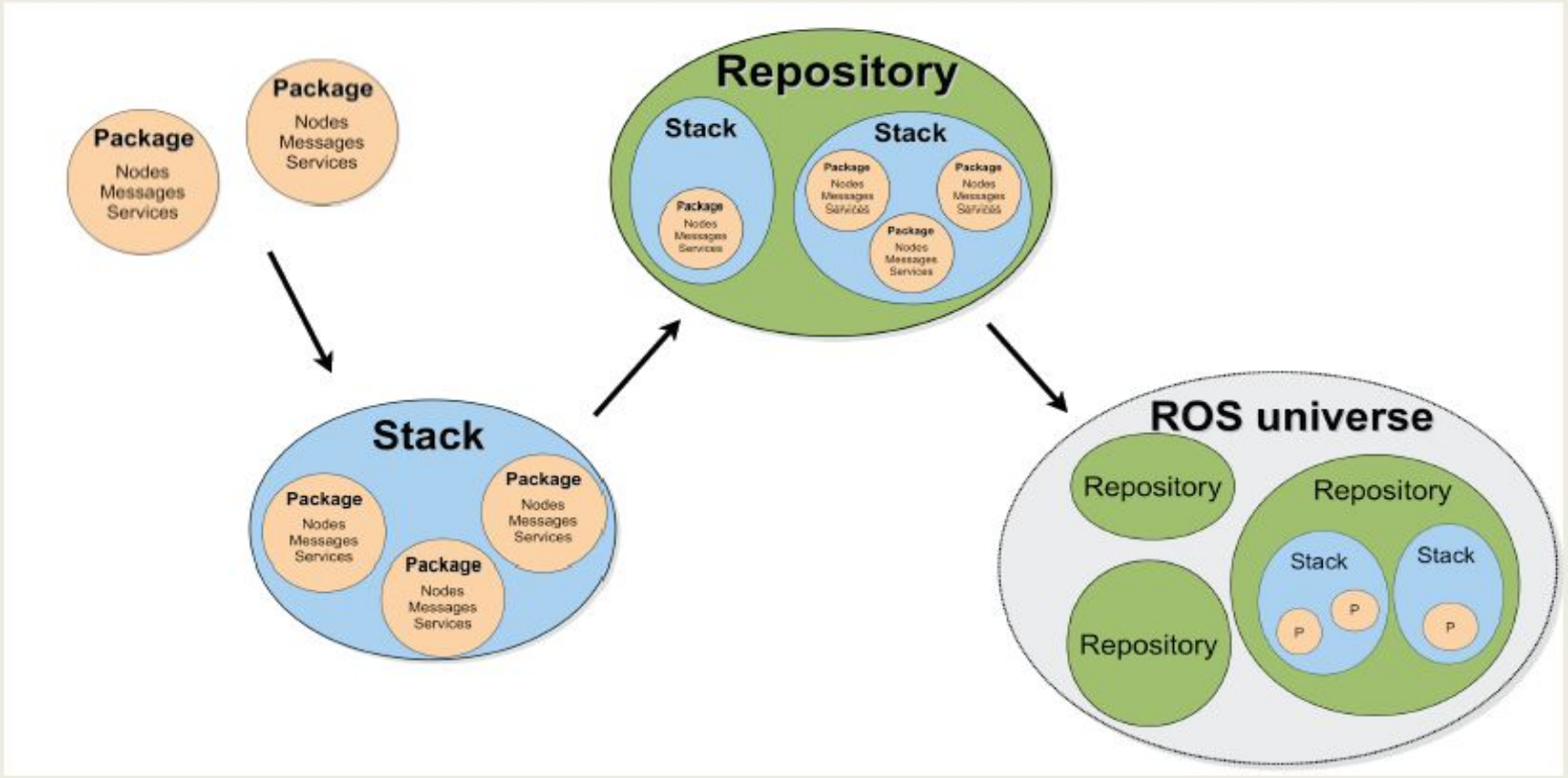
Publisher

```
int main(int argc, char **argv){
  ros::init(argc, argv, "talker");
  ros::NodeHandle n;
  ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
  ros::Rate loop_rate(10);

  while (ros::ok()) {
    std_msgs::String msg;
    msg.data = "hello world ";

    chatter_pub.publish(msg);

    ros::spinOnce();
    loop_rate.sleep();
  }
```

Ros Packages (Пакеты)

- ПО ROS организовано в виде пакетов
- Пакет может содержать в себе код узла (N), код библиотек, определение сообщений, сервисов, файлов запуска и настроек
- Зависимости между пакетами
- Могут объединяться в стеки
- Сборка пакетов одной командой

ROS packages



Taken from Sachin Chitta and Radu Rusu (Willow Garage)

Установка ROS

- Ubuntu
  http://wiki.ros.org/indigo/Installation/Ubuntu
- виртуальная машина
  - http://wiki.ros.org/indigo/Installation

Состав пакета

- src - исходные файлы узлов и библиотек
- include - заголовочные файлы, которые могут быть использованы в других пакетах
- launch - файлы запуска узлов пакета
- python - исходные файлы узлов
- package.xml - файл со описанием проекта
- CMakeList.txt - файл cmake с правилами сборки проекта

package.xml

CMakeList.txt

Catkin cmake

- папки рабочего пространства
  - src - исходные коды пакетов
  - build - промежуточные файлы сборки cmake
  - devel - результаты сборки
  - install - устанавливаемые результаты сборки

ROS environment

- поддержка возможности разработки для разных версий ROS и для разный наборов пакетов
- Каждый раз для запуска приложений ROS необходимо
  - проинициализировать окружение ROS:
    - `$source /opt/ros/indigo/setup.bash`
  - проинициализировать пользовательское окружение (workspace)
    - `$source ~/workspace/ros_ws/devel/setup.bash`
  - `проверка`
    - `$echo $ROS_PACKAGE_PATH`

Использование IDE

- Генерация файлов проекта для IDE Eclipse

  ```
  cd workspace/build/имя_пакета
  cmake ../../src/имя_пакета -DCATKIN_DEVEL_PREFIX=../../devel -G"Eclipse CDT4 - Unix Makefiles"
  ```

- QtCreator IDE

  указать в качестве build каталога
  workspace/build/имя_пакета
  cmake arguments: -DCATKIN_DEVEL_PREFIX=../../devel

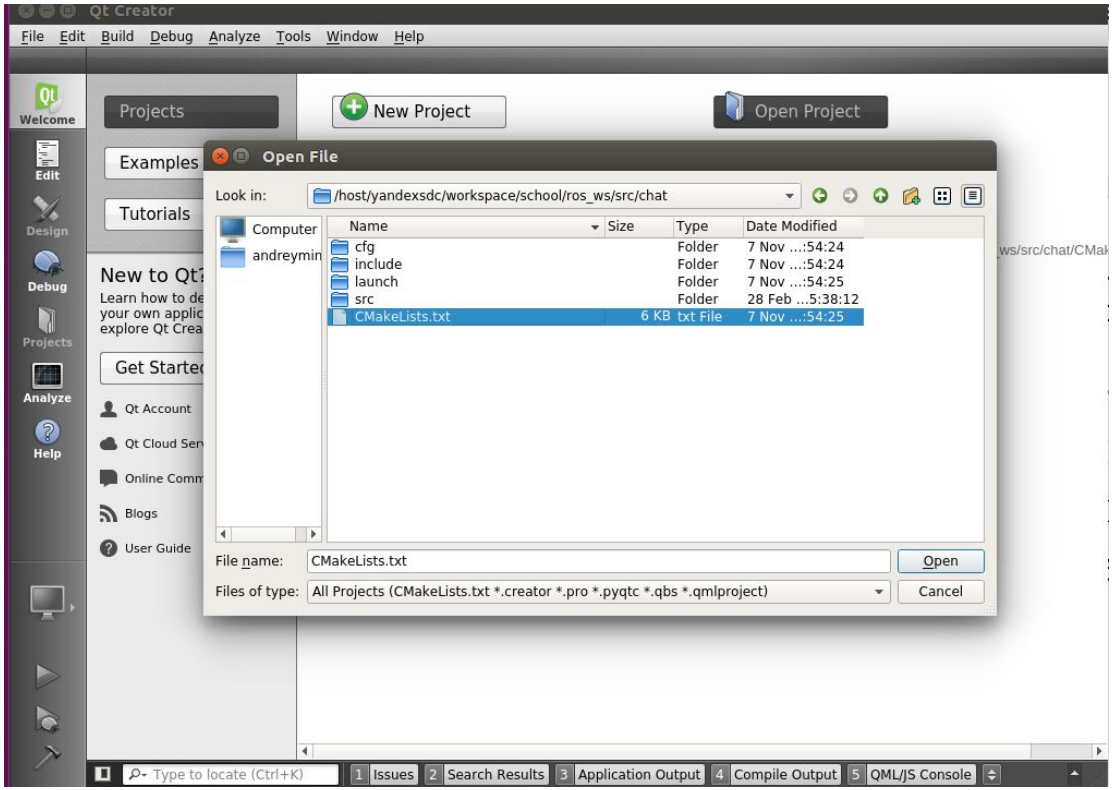
- Запускать IDE из консоли после инициализации workspace($source devel/setup.bash)

Настройка qtcreator

- Установка
  - sudo apt update
  - sudo apt install qtcreator qt5-default

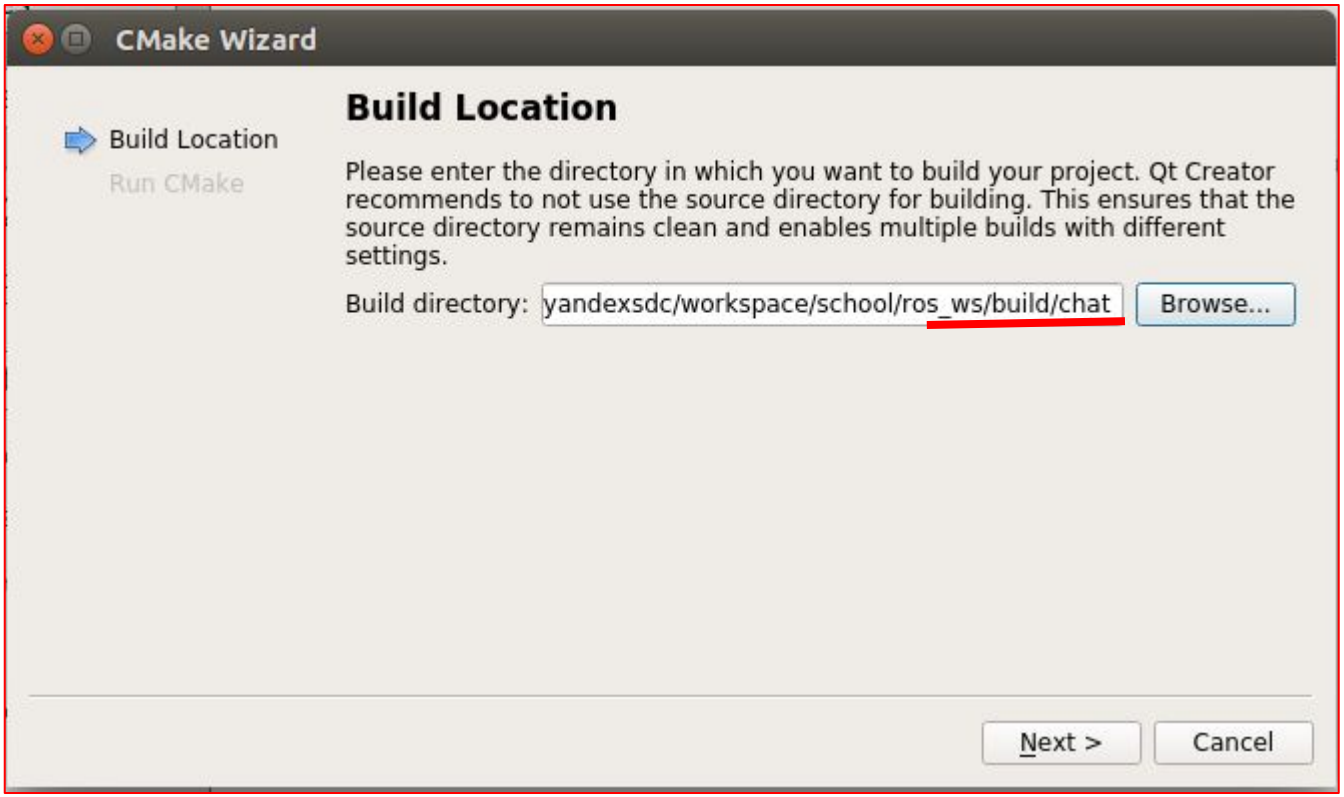
Настройка проекта qtcreator

- Открываем существующий проект
- В качестве файла проекта выбираем CMakeList.txt нашего пакета



Настройка проекта qtcreator

- Указываем директорию построения - build/<имя_пакета>



Настройка проекта qtcreator

- Указываем аргумент cmake - директорию, куда будут помещены исполняемые файлы - devel