ROS

ITMO more than a UNIVERSITY

Операционная система ROS

Введение в основы разработки
программных решений для роботов

О себе

Islam Bzhikhatlov

MSc in engineering, PhD in systems analysis and control

Associated Professor

and doing some projects at ITMO

E-mail: bia@itmo.ru

https://t.me/likerobotics

Содержание курса

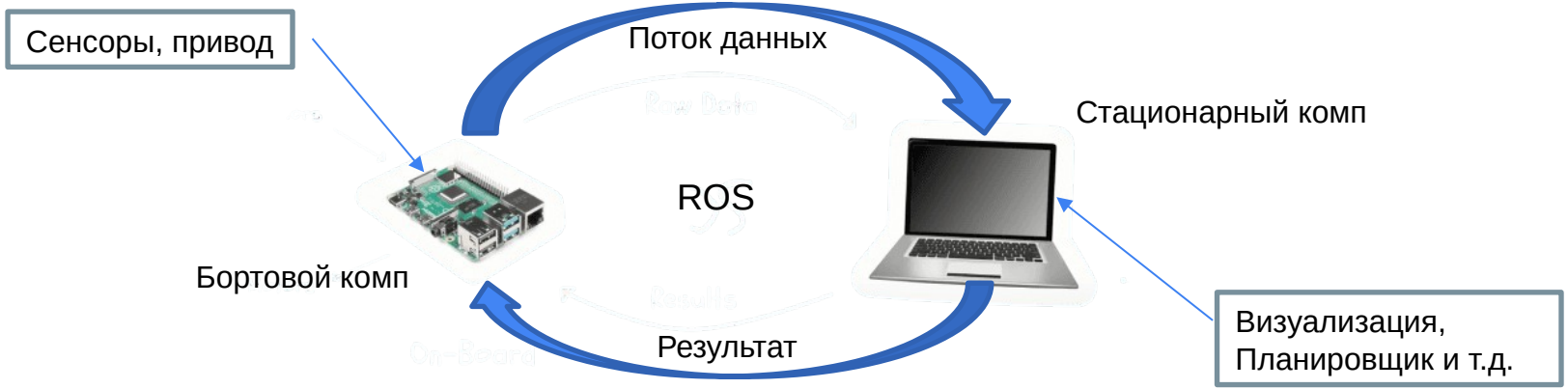
- Архитектура ROS
  - Основы работы с Linux
  - Термины и определения
  - Философия ROS
  - Установка и настройка окружения ROS
  - Структура пакета ROS
  - Изучение готового ROS пакета
- Создание ROS пакетов
  - Создание пакета, ноды, топиков
  - Организация обмена данными
  - Особенности работы с компилятором
  - Типы сообщений, кастомные сообщения
  - Сервисы и действия
  - ROS params
- Модель робота (Gazebo)
  - TF, Rviz, RQT
  - Создание виртуальной модели робота (URDF, Xacro)
  - Создание симуляции в Gazebo (sdf, world)
  - Добавление сенсоров и работа с данными сенсоров
  - Управление движением робота
- Переход из виртуального мира в реальный

Что такое ROS?

ROS - это фреймворк (middleware OS), его можно отнести к системному программному обеспечению, обеспечивающая взаимодействие базовой операционной системы и прикладного программного обеспечения.

Базовые элементы робототехнических систем уже реализованы и доступны в виде пакетов, которые могут дополняться прочими программами в зависимости от задач.

Знание OS Linux является обязательным требованием!



Как устроен ROS

Реализует:

- аппаратная абстракция
- низкоуровневое управление устройствами
- реализация часто используемых функций
- передача сообщений между процессами
- управление пакетами, включая инструменты и библиотеки для получения, сборки, написания и запуска кода на нескольких компьютерах.

ROS имеет три концептуальных уровня : **уровень файловой системы**, **уровень вычислительного графа** и уровень сообщества.

Структура ROS

Уровень файловой системы

- **Packages** – Пакеты — это основная единица организации программного обеспечения в ROS. (Бывают еще и Metapackages) Пакет может содержать процессы выполнения ROS (узлы), ROS-зависимую библиотеку, наборы данных, файлы конфигурации или что-либо еще, что удобно организовано вместе. Пакеты — это самый атомарный элемент сборки.
- Package Manifests: Манифест (**package.xml**) – предоставить метаданные о пакете, включая его имя, версию, описание, зависимости и другие данные, например **об экспортированных пакетах**.
- Message (msg) types (типы сообщений, также бывают типы сервисов) - определяют структуры данных для сообщений в ROS.

Уровень вычислительного графа

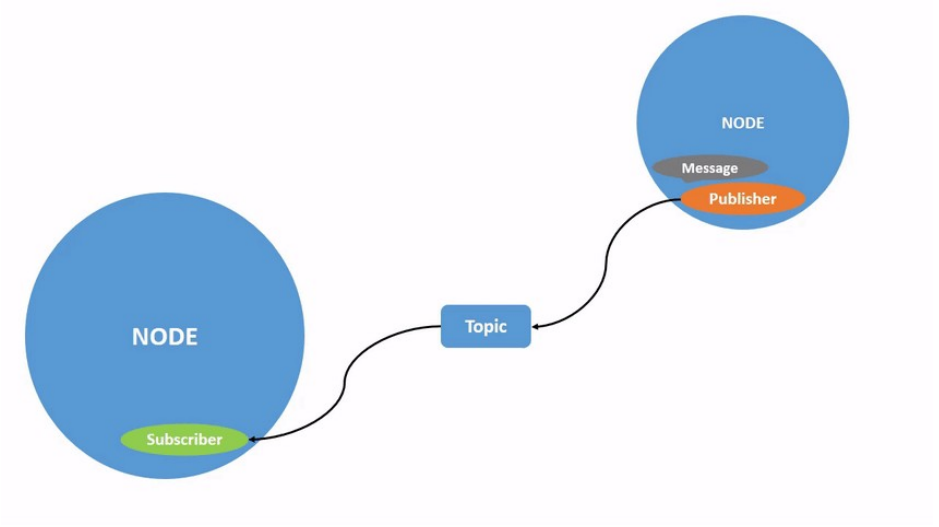
- **Nodes** - это процессы, выполняющие вычисления. ROS спроектирован так, чтобы быть модульным и мелкомасштабным; Обычно система состоит из огромного количества узлов.
- **Master** - обеспечивает регистрацию имени и поиск по остальной части вычислительного графа. Без Мастера узлы не смогут находить друг друга, обмениваться сообщениями или вызывать сервисы. Parameter Server (часть мастера) – позволяет хранить данные централизованно.
- **Messages** - Узлы общаются друг с другом посредством передачи сообщений. Сообщение — это просто структура данных, содержащая типизированные поля. Поддерживаются стандартные примитивные типы (целые, с плавающей запятой, логические значения и т. д.), а также массивы примитивных типов.
- **Topics** – Обмен сообщения происходит через топики. Узел отправляет сообщение, публикуя его в заданном топике. Топик - это имя, которое используется для идентификации содержимого сообщения. Узел, заинтересованный в определенном виде данных, подпишется на соответствующую тему.
- Services – альтернатива topic/subscriber когда нужно взаимодействовать в режиме запрос/ответ.
- Bags - это формат для сохранения и воспроизведения данных сообщений ROS.

Уровень сообщества

- Distributions (например: noetic) – упрощает установку пакетов и согласование
- Repositories
- ROS Wiki

Пример

Самый простой случай обмена данными между узлами!



В рамках курса:

- Версия **ROS Noetic Ninjemys (2020) ROS1**
- Базовая операционная система **Ubuntu 20.04 LTS (** https://releases.ubuntu.com/focal/**)**
- Работа с виртуальной машиной (virtualbox)
- Работа с git
- Основы Linux (Базовые команды)
- Язык программирования Python

Основы git

Git – популярная система контроля версий

Наиболее популярные git площадки:

- Github
- Bitbucket
- gitlab

Основные команды:

git clone

git pull

git push

git add

git commit

Основы Linux

В системе имеется разделение прав пользователя:
- sudo
- owner

Всего 3 вида разрешений:
R – чтение
W – запись
X - исполнение

Как поменять права доступа?
- утилита chmod служит для изменения прав доступа

Например сделать файл исполняемым:
chmod +x file.name

Изменять права доступа может суперпользователь или владелец.

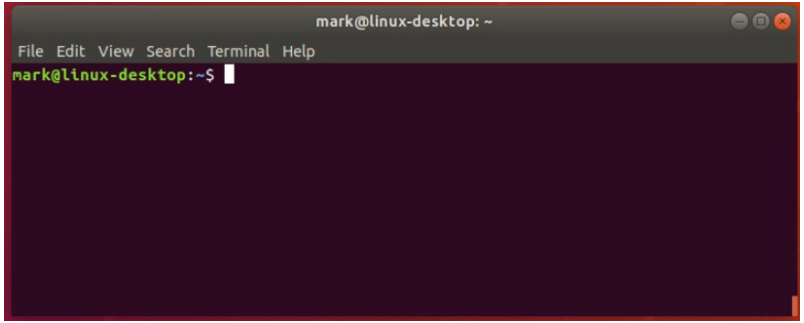
Посмотреть права доступа для директории: **ls -ld**

.bash скрипты – в двух словах.

Сценарии командной строки — это наборы тех же самых команд, которые можно вводить с клавиатуры, собранные в файлы и объединённые некоей общей целью.

Пример: pwd ; whoami

Командная строка (терминал)



Терминал - создан выполнять текстовые команды.

- `CTRL+Alt+T` - открыть терминал
- `CTRL+Shift+T` - новая вкладка в терминале
- `CTRL+Shift+V` - вставить в терминал
- `CTRL+Shift+C` - скопировать в терминале
- `CTRL+C` - вызвать прерывание принудительно
- `mkdir directory_name` - создать папку
- `touch newfile_name` - создать пустой файл
- `TAB TAB` - автопродолжение команды или показать варианты(если их несколько)
- `cd` - смена директории (папки)
- `ls` - показать список содержимого текущей папки (или просто l)
- `ll` - список содержимого текущей папки с детальной информацией (скрытые тоже)
- `tree` - показать содержимое папки в виде дерева

- `pwd` -показать путь до рабочего директория
- `pwd --help` - показывает описание команды
- `history` - история действий в консоли (текущей баш сессии)
- `whereis code` - пути, используемые программой
- `env` - переменные окружения, доступные в текущей сессии
- `grep` - искать внутри файлов (env | grep ROS)
- `~` - корневая директория пользовател (home/user_name)
- `.bashrc` - конфигурационный файл каждой новой bash сессии
- `source .bashrc` - перезагружка конфига в текущую bash сессию

Настройка окружения

- Немного про **Catkin (Catkin Command Line Tools)**

Набор утилит:
- create
- build
- config
- clean
- init

```
apt-get install python3-catkin-tools
mkdir ~/catkin_ws/src
cd ~/catkin_ws
catkin build
source ~/catkin_ws/devel/setup.bash
catkin clean
```

Добавить source /opt/ros/noetic/setup.bash в конфигурационный файл .bashsrc

команда для этого
> echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
> source ~/.bashrc

В директории окружения можно редактировать только содержимое папки src !!!

Программный код

- Для написания прикладных программ рекомендуется использовать редактор Visual Studio Code
- Расширения для VS code: python, cmake, ROS

Структура ROS пакета:

- src
- scripts
- Package.xml
- cmakelist.txt

Как писать меньше кода?

Для этого нужно уметь находить готовые пакеты. Где же искать эти пакеты?

Основной источник: github, сайты производителей устройств

Что-то может не заработать – это нормально!
Ошибки в консоли – это нормально!

ITMO re than a UNIVERSITY

И помните!

Самостоятельная работа студента в рамках курса является обязательной!