CENG 5435/4391: Robotics and ROS AGENDA 9_19_2022

REMIND ME TO RECORD SESSIONS

Blackboard/zoom Session

Contents

Agenda 9_19_2022

Introducing iRobot's platform for potential

https://www.facebook.com/iRobotEducation/videos/780994056545209/?extid=NS-UNK-UNK-UNK-IOS_GK0T-GK1C

Meet the Create® 3 educational robot. iRobot's new mobile development platform for learning ROS 2 and Python.

HOMEWORK 2 REVIEW
1_HW2_5435_4391.pdf

1a_HW2_ANS_5435_4391_Fall2022.pdf

1b_HW2_Q1_Kat_ANSPaganM_CENG5435.docx

Linux Review
2a_UbuntuBasicTutorial_9_12_2022.pdf

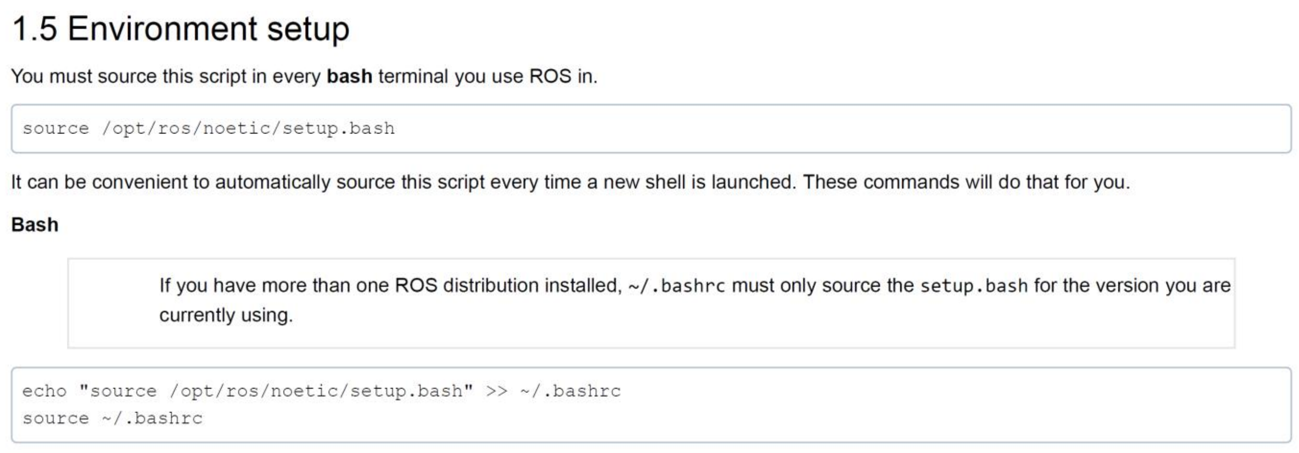
2b__Linux_Term_Cmd_Linux TV_9_09_2022C.pdf

CHAPTER1 IN TEXTBOOK AND TURTLESIM
3_Turtlesim Demo   See Chapter 1 of our Book – updated for Noetic

Textbook for Course *RosRoboticsByExample*   Text

http://wiki.ros.org/noetic/Installation/Ubuntu

(Page 5-8 in textbook for Kinetic)



1.5 Environment setup

You must source this script in every **bash** terminal you use ROS in.

```
source /opt/ros/noetic/setup.bash
```

It can be convenient to automatically source this script every time a new shell is launched. These commands will do that for you.

Bash

If you have more than one ROS distribution installed, `~/.bashrc` must only source the `setup.bash` for the version you are currently using.

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

3b_Bashrc_9_12_2022.pdf   **Show the .bashrc and source alias**

Alias foxy or noetic
harman@harman-VirtualBox:~$ **noetic**
ROS_DISTRO was set to 'foxy' before. Please make sure that the

```
        environment does not mix paths from different distributions.
harman@harman-VirtualBox:~$ env | grep ROS
        ROS_VERSION=1
        ROS_PYTHON_VERSION=3
        ROS_PACKAGE_PATH=/opt/ros/noetic/share
        ROSLISP_PACKAGE_DIRECTORIES=
        ROS_DOMAIN_ID=231
        ROS_ETC_DIR=/opt/ros/noetic/etc/ros
        ROS_MASTER_URI=http://localhost:11311
        ROS_LOCALHOST_ONLY=0
        ROS_ROOT=/opt/ros/noetic/share/ros
        ROS_DISTRO=noetic
```

CATKIN WORKSPACE PAGES 9-10

The next step is to create a catkin workspace. A catkin workspace is a directory (folder) in which you can create or modify existing catkin packages. The catkin structure simplifies the build and installation process for your ROS packages. The ROS wiki website is `http://wiki.ros.org/catkin/Tutorials/create_a_workspace`.

A catkin workspace can contain up to three or more different subdirectories (`/build`, `/devel`, and `/src`), each of which serve a different role in the software development process.

We will label our catkin workspace `catkin_ws`. To create the catkin workspace, typethe following commands:

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

Even though the workspace is empty (there are no packages in the `src` folder, just a single `CMakeLists.txt` link), you can still build the workspace by typing the following commands:

```
$ cd ~/catkin_ws/
$ catkin_make
```

The `catkin_make` command creates the catkin workspace. If you view your current directory contents, you should now have the `build` and `devel` folders. Inside the `devel` folder there are now several `setup.*sh` files. We will source the `setup.bash`file to overlay this workspace on top of your ROS environment:

```
$ source ~/catkin_ws/devel/setup.bash
```

Remember to add this source command to your `.bashrc` file by typing the following command:

```
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

To make sure your workspace is properly overlaid by the setup script, make sure the `ROS_PACKAGE_PATH` environment variable includes the directory you're in by typingthe following command:

```
$ echo $ROS_PACKAGE_PATH
```

The output of the preceding command should be as follows:

```
    /home/<username>/catkin_ws/src:/opt/ros/kinetic/share
```

Here, `<username>` is the name you chose for the user when Ubuntu was installed.

```
harman@harman-VirtualBox:~$ echo $ROS_PACKAGE_PATH
      /opt/ros/noetic/share
```

AT EACH TERMINAL IF NOT IN .bashrc

```
harman@harman-VirtualBox:~$ source ~/catkin_ws/devel/setup.bash
harman@harman-VirtualBox:~$ echo $ROS_PACKAGE_PATH
  /home/harman/catkin_ws/src:/opt/ros/noetic/share
```

Let's find the package.xml for noetic

XX harman@harman-VirtualBox:/opt/ros/noetic$ **locate package.xml**

(Many many pages – a few examples:

```
        /opt/ros/noetic/share/turtle_actionlib/package.xml
        /opt/ros/noetic/share/turtle_tf/package.xml
        /opt/ros/noetic/share/turtle_tf2/package.xml
        /opt/ros/noetic/share/turtlesim/package.xml
```

```
harman@harman-VirtualBox:/opt/ros/noetic/share$
harman@harman-VirtualBox:/opt/ros/noetic/share$ cd turtlesim
harman@harman-VirtualBox:/opt/ros/noetic/share/turtlesim$ ls
 cmake  images  msg  package.xml  srv
harman@harman-VirtualBox:/opt/ros/noetic/share/turtlesim$ gedit package.xml
```

REMEMBER – YOU DID NOT HAVE TO TYPE THIS

```
        <?xml version="1.0"?>
        <package>
```

```
 <name>turtlesim</name>
 <version>0.10.2</version>
 <description>
   turtlesim is a tool made for teaching ROS and ROS packages.
 </description>
 <maintainer email="dthomas@osrfoundation.org">Dirk Thomas</maintainer>
 <license>BSD</license>

 <url type="website">http://www.ros.org/wiki/turtlesim</url>
 <url type="bugtracker">https://github.com/ros/ros_tutorials/issues</url>
 <url type="repository">https://github.com/ros/ros_tutorials</url>
 <author>Josh Faust</author>

 <buildtool_depend>catkin</buildtool_depend>

 <build_depend>geometry_msgs</build_depend>
 <build_depend>libboost-thread-dev</build_depend>
 <build_depend>qtbase5-dev</build_depend>
 <build_depend>message_generation</build_depend>
 <build_depend>qt5-qmake</build_depend>
 <build_depend>rosconsole</build_depend>
 <build_depend>roscpp</build_depend>
 <build_depend>roscpp_serialization</build_depend>
 <build_depend>roslib</build_depend>
 <build_depend>rostime</build_depend>
 <build_depend>std_msgs</build_depend>
 <build_depend>std_srvs</build_depend>

 <run_depend>geometry_msgs</run_depend>
 <run_depend>libboost-thread-dev</run_depend>
 <run_depend>libqt5-core</run_depend>
 <run_depend>libqt5-gui</run_depend>
 <run_depend>message_runtime</run_depend>
 <run_depend>rosconsole</run_depend>
 <run_depend>roscpp</run_depend>
 <run_depend>roscpp_serialization</run_depend>
 <run_depend>roslib</run_depend>
 <run_depend>rostime</run_depend>
 <run_depend>std_msgs</run_depend>
 <run_depend>std_srvs</run_depend>
</package>
```

```
harman@harman-VirtualBox:/opt/ros/noetic/share/turtlesim/msg$ ls
        Color.msg  Pose.msg
harman@harman-VirtualBox:/opt/ros/noetic/share/turtlesim/msg$ gedit Pose.msg
        float32 x
        float32 y
        float32 theta

        float32 linear_velocity
```

float32 angular_velocity

LET'S GO THROUGH THE BOOK WITH NOETIC FOR REAL!

Noetic means intellect.

As ROS's developer Open Robotics describe Noetic, "there is perhaps no better way to describe the entire pursuit of ROS 1".

For **Ninjemys**, it is **an extinct turtle** species in Australia, according to Wikipedia. The Ninjemys turtles have a large pair of horns on its head stuck out to the sides.
https://varhowto.com/ros-noetic/

PAGES 11-32 in ROS Robotics By Example 2nd
Fire up Virtual Box and Noetic!

Text Pages 11-19 General
5a_Chapter1_9_19_2022_ToPage19.txt

TurtlesimDemo_9_19_22

5c_ROS_TsimInfo_Run_9_19_2022.txt

RECAP **OF TURTLESIM CLI COMMANDS: (PAGE 32 IN TEXT)**

ROS commands summary

If you are communicating with ROS via the terminal window, it is possible to issuecommands to ROS to explore or control nodes in a package from the command prompt, as listed in the following table:

Command	**Action**	**Example usage and subcommand examples**
`roscore`	Starts the Master	`$ roscore`
`rosrun`	Runs an executable program and creates nodes	`$ rosrun [package name] [executable name]`

`rosnode`	Shows information about nodes and lists the active nodes	`$ rosnode info [node name]`
		`$ rosnode<subcommand>`
		Subcommand: `list`

`rostopic`	Shows information about ROS topics	`$ rostopic<subcommand><topic name>` Subcommands: `echo`, `info`, and `type`
`rosmsg`	Shows information about the message types	`$ rosmsg<subcommand> [package name]/ [message type]` Subcommands: `show`, `type`, and `list`
`rosservice`	Displays the runtime information about various services and allows the display of messages being sent to a topic	`$ rosservice<subcommand> [service name]` Subcommands: `args`, `call`, `find`, `info`, `list`, and `type`
`rosparam`	Used to get and set parameters (data) used by nodes	`$ rosparam<subcommand> [parameter]` Subcommands: `get`, `set`, `list`, and `delete`

The website (`http://wiki.ros.org/ROS/CommandLineTools`) describes many ROScommands. The table lists some important ones. However, these examples only cover a few of the possible variations of the commands.

http://wiki.ros.org/ROS/CommandLineTools

COMMANDS CHAPTER 1

Chapter1_9_19_2022_ToPage19.txt