5435_4391 AGENDA 11/21/2022 11/21/2022 3:42:09 PM

Contents

START LECTURE 11/21/2022 RECORD

Class Rules:

For CENG 5435_4391 Fall 2022

1. If you scored Below 80 on the Quiz, you will have a Final Exam in class. The exam will cover all the lectures and material from the class.
2. If below 70, you can redo the exam for more points with the conditions:
   a. **Give references for your answers from the class lectures or from data on the websites.**
   b. You must come to D158 and complete some programming assignments.

November 28, 2022 Brief Lecture and and Final Exam for some- 1HR Closed Book in Class

Final Night December 5 - Final Reports on Robots

Course Evaluation

https://apps.uhcl.edu/OnlineEvals/Auth/Login?ReturnUrl=%2fOnlineEvals

https://apps.uhcl.edu/OnlineEvals

Lecture November 21, 2022

I. **1_ROS1_SummaryWhatIsROS_MostlyROS1.pdf**

2_turtlesim_py_1 gotogoal 10_13_2022.pdf

II. **An 18-month journey to TurtleBot 4.mp4** 18:24 3_ROSCon22 TurtleBot4a.pdf
https://vimeo.com/showcase/9954564/video/767165480

III. ROS2_SummarySlides

4_ROS2_SummarySlides.pptx

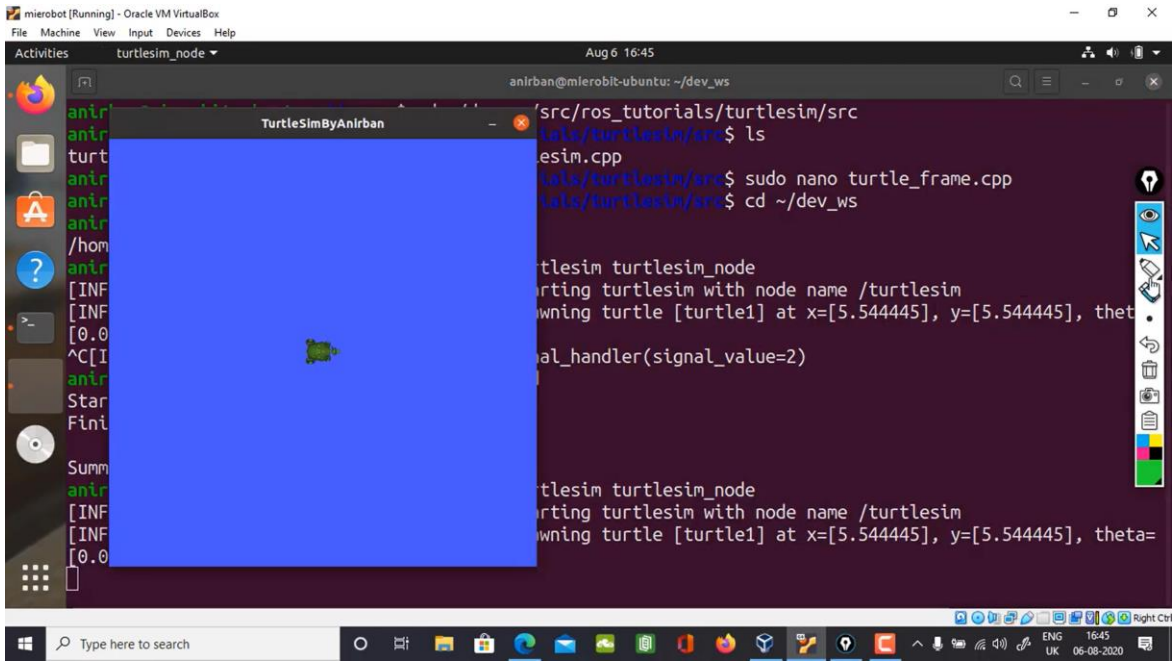
5 _ROS_DATA&INFORMATION_F22.pdf

6_ **Ros1_ROS2FeaturesPowerPoint Presentation.pdf**

ROS2 Tutorials- Workspace and Colcon Build with Turtle bot Turtlesim – Rename Screen

Creating a workspace ForMieRoboticsROS 2_Foxy.pdf

https://www.youtube.com/watch?v=goLXowdo1PE&t=81s



7 Modify the overlay

You can modify `turtlesim` in your overlay by editing the title bar on the turtlesim window. To do this, locate the `turtle_frame.cpp` file in `~/ros2_ws/src/ros_tutorials/turtlesim/src`. Open `turtle_frame.cpp` with your preferred text editor.

On line 52 you will see the function `setWindowTitle("TurtleSim");`. Change the value `"TurtleSim"` to `"MyTurtleSim"`, and save the file.

References for this video:

https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html

Need Workspace to use

Alias foxy or noetic

harman@harman-VirtualBox:~$ foxy

harman@harman-VirtualBox:~$ cd ~/ros2_ws/src/ros_tutorials/turtlesim/src

bash: cd: /home/harman/ros2_ws/src/ros_tutorials/turtlesim/src: No such file or directory

harman@harman-VirtualBox:~$ cd ros2_ws

bash: cd: ros2_ws: No such file or directory

harman@harman-VirtualBox:~$

ROS2 Basics #11 - Writing a Simple Publisher and Subscriber (Python)

https://www.youtube.com/watch?v=eqfoy2ctixE&t=169s

TutorialsROS 2_ Foxy documentation.pdf

ROS 2 Cheats Sheet

Command Line Interface

All ROS 2 CLI tools start with the prefix 'ros2' followed by a command, a verb and (possibly) positional/optional arguments.
For any tool, the documentation is accessible with,

```
$ ros2 command --help
```

and similarly for verb documentation,

```
$ ros2 command verb -h
```

Similarly, auto-completion is available for all commands/verbs and most positional/optional arguments. E.g.,

```
$ ros2 command [tab][tab]
```

Some of the examples below rely on:
ROS 2 demos package.

---

action Allows to manually send a goal and displays debugging information about actions.
Verbs:

info	Output information about an action.
list	Output a list of action names.
send_goal	Send an action goal.
show	Output the action definition.

Examples:

```
$ ros2 action info /fibonacci
$ ros2 action list
$ ros2 action send_goal /fibonacci \
 action_tutorials/action/Fibonacci "order: 5"
$ ros2 action show action_tutorials/action/Fibonacci
```

---

bag Allows to record/play topics to/from a rosbag.
Verbs:

info	Output information of a bag.
play	Play a bag.
record	Record a bag.

Examples:

```
$ ros2 info <bag-name>
$ ros2 play <bag-name>
$ ros2 record -a
```

---

component Various component related verbs.
Verbs:

list	Output a list of running containers and components.
load	Load a component into a container node.
standalone	Run a component into its own standalone container node.
types	Output a list of components registered in the ament index.
unload	Unload a component from a container node.

Examples:

```
$ ros2 component list
$ ros2 component load /ComponentManager \
 composition composition::Talker
$ ros2 component types
$ ros2 component unload /ComponentManager 1
```

---

daemon Various daemon related verbs.
Verbs:

start	Start the daemon if it isn't running.
status	Output the status of the daemon.
stop	Stop the daemon if it is running

---

doctor A tool to check ROS setup and other potential issues such as network, package versions, rmw middleware etc.
Alias: **wtf** (where's the fire).
Arguments:

--report/-r	Output report of all checks.
--report-fail/-rf	Output report of failed checks only.
--include-warning/-iw	Include warnings as failed checks.

Examples:

```
$ ros2 doctor
$ ros2 doctor --report
$ ros2 doctor --report-fail
$ ros2 doctor --include-warning
$ ros2 doctor --include-warning --report-fail
```

or similarly,

```
$ ros2 wtf
```

---

extension_points List extension points.

---

extensions List extensions.

---

interface Various ROS interfaces (actions/topics/services)-related verbs. Interface type can be filtered with either of the following option, '--only-actions', '--only-msgs', '--only-srvs'.
Verbs:

list	List all interface types available.
package	Output a list of available interface types within one package.
packages	Output a list of packages that provide interfaces.
proto	Print the prototype (body) of an interfaces.
show	Output the interface definition.

Examples:

```
$ ros2 interface list
$ ros2 interface package std_msgs
$ ros2 interface packages --only-msgs
$ ros2 interface proto example_interfaces/srv/AddTwoInts
$ ros2 interface show geometry_msgs/msg/Pose
```

---

launch Allows to run a launch file in an arbitrary package without to 'cd' there first.
Usage:

```
$ ros2 launch <package> <launch-file>
```

Example:

```
$ ros2 launch demo_nodes_cpp add_two_ints.launch.py
```

---

lifecycle Various lifecycle related verbs.
Verbs:

get	Get lifecycle state for one or more nodes.
list	Output a list of available transitions.
nodes	Output a list of nodes with lifecycle.
set	Trigger lifecycle state transition.

---

msg (deprecated) Displays debugging information about messages.
Verbs:

list	Output a list of message types.
package	Output a list of message types within a given package.
packages	Output a list of packages which contain messages.
show	Output the message definition.

Examples:

```
$ ros2 msg packages
$ ros2 msg show geometry_msgs/msg/Pose
```

---

multicast Various multicast related verbs.

Verbs:

receive	Receive a single UDP multicast packet.
send	Send a single UDP multicast packet.

---

node Displays debugging information about nodes.

Verbs:

info	Output information about a node.
list	Output a list of available nodes.

Examples:

```
$ ros2 node info /talker
$ ros2 node list
```

---

param Allows to manipulate parameters.

Verbs:

delete	Delete parameter.
describe	Show descriptive information about declared parameters.
dump	Dump the parameters of a given node in yaml format, either in terminal or in a file.
get	Get parameter.
list	Output a list of available parameters.
set	Set parameter

Examples:

```
$ ros2 param delete /talker /use_sim_time
$ ros2 param get /talker /use_sim_time
$ ros2 param list
$ ros2 param set /talker /use_sim_time false
```

---

pkg Create a ros2 package or output package(s)-related information.

Verbs:

create	Create a new ROS2 package.
executables	Output a list of package specific executables.
list	Output a list of available packages.
prefix	Output the prefix path of a package.
xml	Output the information contained in the package xml manifest.

Examples:

```
$ ros2 pkg prefix std_msgs
$ ros2 pkg xml -t version
```

---

run Allows to run an executable in an arbitrary package without having to 'cd' there first.

Usage:

```
$ ros2 run <package> <executable>
```

Example:

```
$ ros2 run demo_node_cpp talker
```

---

security Various security related verbs.

Verbs:

create_key	Create key.
create_permission	Create keystore.
generate_artifacts	Create permission.
list_keys	Distribute key.
create_keystore	Generate keys and permission files from a list of identities and policy files.
distribute_key	Generate XML policy file from ROS graph data.
generate_policy	List keys.

Examples (see sros2 package):

```
$ ros2 security create_key demo_keys /talker
$ ros2 security create_permission demo_keys /talker \
 policies/sample_policy.xml
$ ros2 security generate_artifacts
$ ros2 security create_keystore demo_keys
```

---

service Allows to manually call a service and displays debugging information about services.

Verbs:

call	Call a service.
find	Output a list of services of a given type.
list	Output a list of service names.
type	Output service's type.

Examples:

```
$ ros2 service call /add_two_ints \
 example_interfaces/AddTwoInts "a: 1, b: 2"
$ ros2 service find rcl_interfaces/srv/ListParameters
$ ros2 service list
$ ros2 service type /talker/describe_parameters
```


Verbs:

list	Output a list of available service types.
package	Output a list of available service types within one package.
packages	Output a list of packages which contain services.
show	Output the service definition.

---

test Run a ROS2 launch test.

---

topic A tool for displaying debug information about ROS topics, including publishers, subscribers, publishing rate, and messages.

Verbs:

bw	Display bandwidth used by topic.
delay	Display delay of topic from timestamp in header.
echo	Output messages of a given topic to screen.
find	Find topics of a given type type.
hz	Display publishing rate of topic.
info	Output information about a given topic.
list	Output list of active topics.
pub	Publish data to a topic.
type	Output topic's type.

Examples:

```
$ ros2 topic bw /chatter
$ ros2 topic echo /chatter
$ ros2 topic find rcl_interfaces/msg/Log
$ ros2 topic hz /chatter
$ ros2 topic info /chatter
$ ros2 topic list
$ ros2 topic pub /chatter std_msgs/msg/String \
 'data: Hello ROS 2 world'
$ ros2 topic type /rosout
```