ROS Indigo Cheatsheet

Filesystem Management Tools

- **rospack**: A tool for inspecting packages.
- **rospack profile**: Prints the profile information for a package.
- **roscd**: Change to the package's directory.
- **rosnode**: List nodes in a package.
- **roscore**: Start or stop the ROS core.
- **rosout**: Open ROS log files in a text editor.
- **roscp**: Copy a file from one place to another.
- **roslisp**: Compile a Lisp file.
- **rosnode**: Install package system dependencies.
- **roswait**: Displays error messages about a running ROS service or launch file.
- **catkin absolutely path to workspace**: Builds a ROS catkin workspace.
- **wstool**: Manages workspace dependencies.
- **rqt-dep**: Displays package structure and dependencies.

Usage:
```bash
$ rospack find [package]
$ rospack [package/subdir] + * + -
$ rqt
$ roscd [package/subdir]
$ rosnode [package/subdir]
$ rosrun [package] [file]
$ rospack [package] [destination]
$ rostopic [package] [topic]
$ rqt [package] [topic]
$ catkin absolutely path to workspace /rqt-dep [options]
```

Start-up and Process Launch Tools

**rosmake**

The basic nodes and programs for ROS-based systems. A rosmake must be running for ROS nodes to communicate.

Usage:
```bash
$ rosmake
$ rosversion
```

Starts a rosmake with minimal typing. Usage:
```bash
$ rosmake [package/executable]
```

Example (runs turbine): `$ rosmake turbine_turbine_node`

**roslaunch**

Starts a rosmake (if needed), local nodes, remote nodes via SSH, and sets parameter server connections.

Examples:
- Launch a file in a package: `$ roslaunch package name file name.launch`
- Launch a different port: `$ roslaunch -p 1234 package name file name.launch`
- Launch on the local node: `$ roslaunch --local package name file name.launch`

Logging Tools

**rosbag**

A tool for recording and replaying bag files. Usage:
```bash
$ rosbag record [options] [topic] [node] [rate]
$ rosbag play [options] [topic] [node] [rate]
$ rosbag compress [options] [topic] [node] [rate]
$ rosbag decompress [options] [topic] [node] [rate]
$ rosbag filter [options] [topic] [node] [rate]
```

Examples:
- Record selected topics: `$ rosbag record topic1 topic2`
- Replay all messages without silence: `$ rosbag play -a demo_bag`
- Replay several bag files in sequence: `$ rosbag play demo1.bag demo2.bag`

Introspection and Command Tools

**roslog/rosvr**

Displays Message/Service (log/err) data structures definition. Usage:
```bash
$ rosvr show [options] [node] [topic]
$ rosvr list [options] [node] [topic]
$ rosvr nfind [options] [node] [topic]
$ rosvr package [options] [node] [topic]
$ rosvr session [options] [node] [topic]
```

Examples:
- Display the Pose msg: `$ rosvr show Pose`
- List the messages in the nav_msgs package: `$ rosvr package nav_msgs`
- List the messages using sensor_msgs/CameraInfo: `$ rosvr session sensor_msgs/CameraInfo`

**rosnode**

Displays debugging information about ROS nodes, including publications, subscriptions and connections.

Usage:
```bash
$ rosnodes [options]
```

Examples:
- Test connectivity to node: `$ rosnodes ping <node>`
- List active nodes: `$ rosnodes list`
- Print information about a node: `$ rosnodes info <node>`
- List nodes running on a machine: `$ rosnodes machine <machine>`
- Kill a running node: `$ rosnodes kill <node>`

Examples:
- Kill all nodes: `$ rosnodes kill -a`
- List nodes on a machine: `$ rosnodes machine <machine>`
- Ping all nodes: `$ rosnodes ping -a`

**rostopic**

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

Usage:
```bash
$ rostopic show [options] [node] [topic]
$ rostopic list [options] [node] [topic]
$ rostopic info [options] [node] [topic]
$ rostopic kill [options] [node] [topic]
$ rostopic type [options] [node] [topic]
```

Examples:
- Publish hello at 10 Hz: `$ rostopic pub -r 10 /topic name stdmsgs/String hello`
- Clear the screen after each message is published: `$ rostopic echo -n /topic name`
- Display messages that match a given Python expression: `$ rostopic echo-filter "a.data="foo"/topic name"`

**rostopam**

A tool for getting and setting ROS parameters on the parameter server using YAML-encoded files.

Usage:
```bash
$ rostopam set [options] [node] [topic]
$ rostopam get [options] [node] [topic]
$ rostopam load [options] [node] [topic]
$ rostopam delete [options] [node] [topic]
```

Examples:
- List all parameters in a namespace: `$ rostopam list [namespace]`
- Set a parameter: `$ rostopam set [enum] [namespace] [value]`
- Clear only the parameters in a namespace to a file: `$ rostopam dump_yaml [namespace]`

**rosworld**

A tool for listing and querying ROS services.

Usage:
```bash
$ rosworld [options] [node] [topic]
```

Examples:
- Print information about active services: `$ rosworld list [node] [topic]`
- Name of provider: `$ rosworld name [node] [topic]`
- Call the service with the given args: `$ rosworld call [node] [topic] [args]`
- List the arguments of a service: `$ rosworld arg [node] [topic]`
- Print the service type: `$ rosworld type [node] [topic]`
- Print the service ROSRPC uri: `$ rosworld uri [node] [topic]`
- Find services by service type: `$ rosworld find [node] [service]`

Examples:
- Call a service from the command line: `$ rosworld call /AddTwoInts 1 2`
- Pipe the output of rostopam to rosspy to view the srv type: `$ rosservice type /AddTwoInts`
ROS Indigo Cheatsheet

Logging Tools

**rqt_console**
A tool to display and filtering messages published on a topic.

Usage:
$ rqt_console

**rqt_log**
A tool to stream logging messages into a file.

Usage, viewing:
$ rqt_log logfile.log
Usage, logging:
$ rqt_log -p The big red log button.

**rqt_logger_level**
Change the log level of ROS nodes. This will increase or decrease the information they log to the screen and repositories.

Usage:
viewing $ rqt_logger_level

Introspection & Command Tools

**rqt_topic**
A tool for viewing published topics in real time.

Usage:
$ rqt

**rqt_msg, rqt_srv, and rqt_action**
A tool for viewing available messages, services, and actions.

Usage:
$ rqt

**rqt_publisher, and rqt_service_caller**
Tools for publishing messages and calling services.

Usage:
$ rqt
Plugin Menu->Topic->Message Publisher
Plugin Menu->Service->Service Caller

**rqt_graph, and rqt_dep**
Tools for displaying graphs of running ROS nodes with connecting topics and package dependencies respectively.

Usage:
$ rqt_graph
$ rqt_dep

**rqt_top**
A tool for ROS specific process monitoring.

Usage:
$ rqt
Plugin Menu->Introspection->Process Monitor

**rqt_reconfigure**
A tool for dynamically reconfiguring ROS parameters.

Usage:
$ rqt
Plugin Menu->Configuration->Dynamic Reconfigure

Development Environments

**rqt_shell, and rqt_py_console**
Two tools for accessing an xterm shell and python console respectively.

Usage:
$ rqt
Plugin Menu->Miscellaneous Tools->Shell
Plugin Menu->Miscellaneous Tools->Python Console

Data Visualization Tools

**tf_echo**
A tool that prints the information about a particular transformation between a source frame and a target frame.

Usage:
$ rostopic tf_echo <source_frame> <target_frame>

Examples:
To echo the transform between /map and /odom:
$ rostopic tf_echo /map /odom

**view_frames**
A tool for visualizing the full tree of coordinate transforms.

Usage:
$ roscript tf_tools view_frames.py
$ save_frames.pdf

**rqt_plot**
A tool for plotting data from ROS topic fields.

Usage:
$ rqt_plot

Examples:
To graph the data in different plots:
$ rqt_plot /topic/field1 /topic/field2
To graph the data on the same plot:
$ rqt_plot /topic/field1 /topic/field2
To graph multiple fields of a message:
$ rqt_plot /topic/field1:field2:field3

**rqt_image_view**
A tool to display image topics.

Usage:
$ rqt_image_view

https://robosub.eecs.wsu.edu/wiki/cs/ros/cheatsheet/start

Printed on 2022/02/02 23:15
ROS Indigo Catkin Workspaces

Create a catkin workspace

Setup and use a new catkin workspace from scratch.

Example:
```
$ source /opt/ros/indigo/setup.bash
$ mkdir -p /catkin_ws/src
$ cd /catkin_ws/src
$ catkin init workspace
```

Checkout an existing ROS package

Get a local copy of the code for an existing package and keep it up to date using wstool.

Example:
```
$ cd /catkin_ws/src
$ wstool init
$ wstool set tutorials --git git://github.com/ros/tutorials.git
$ wstool update
```

Create a new catkin ROS package

Create a new ROS catkin package in an existing workspace with catkin create package. After using this you will need to edit the CMakeLists.txt to detail how you want your package built and add information to your package.xml.

Usage:
```
catkin_create_pkg <package_name> [depend1] [depend2]
```

Example:
```
$ cd /catkin_ws/src
$ catkin_create_pkg tutorials std_msgs rospkg roscpp
```

Build all packages in a workspace

Use catkin_make to build all the packages in the workspace and then source the setup.bash to add the workspace to the ROS_PACKAGE_PATH.

Examples:
```
$ cd /catkin_ws
$ catkin_make
$ source devel/setup.bash
```

---

From: https://robosub.eecs.wsu.edu/wiki/ - Palouse RoboSub Technical Documentation

Permanent link: https://robosub.eecs.wsu.edu/wiki/cs/ros/cheatsheet/start

Last update: 2016/11/16 12:34