ROS Indigo Cheatsheet
Filesystem Management Tools

**rosrun**
A tool for inspecting packages.

**rosrun profile**
Finds path and plugin problems.

**roscore**
Change directory to a package.

**rospack**
Publish equivalent for ROS.

**rospack/rospack**
Lists packages or stack information.

**roslaunch**
Runs a ROS launch file.

**rospack**
Copy a file from one place to another.

**rosdep**
Installs package system dependencies.

**rqt**
Displays errors and warnings about a running ROS system or launch file.

**catkin_create_pkg**
Creates a new ROS stack.

**wetool**
Manage many repos in workspace.

**catkin_make**
Builds a ROS catkin workspace.

**rqt_dep**
Displays package structure and dependencies.

Usage:
- `rosrun find [package]`
- `rosrun [package/subdir]` [or `*` or `-`]
- `rosrun`
- `rosrun [package/subdir]`
- `rosrun [package] [file]`
- `rosrun [package] [destination]`
- `rosrun install [package]`
- `rosrun [or rosf] file`
- `catkincreatepkg [package] [depend1]...[dependN]`
- `rqtdep [options]`

* Start-up and Process Launch Tools*

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

Usage:
- `roscfg`

**rosmake**
Runs a ROS package's executable with minimal typing.

Usage:
- `rosmake package executable`

Example (runs tourlilinx): `rosmake tourlilinx tourlilinx_node`

**rosinstall**
Starts a roscore (if needed), local nodes, remote nodes via SSH, and sets parameter server environment.

Usage:
- `rosinstall package file name.launch`
- `rosinstall -p 1234 package file name.launch`
- `rosinstall --local-package file name.launch`

Logging Tools

**roslaunch**
A tool for recording and playing back ROS topics.

Commands:
- `roslaunch` record
- `roslaunch` play
- `roslaunch` compress
- `roslaunch` decompress
- `roslaunch` filter

Examples:
- `roslaunch record topic1 topic2`
- `roslaunch play -a demo_bag`
- `roslaunch compress demo_bag`
- `roslaunch decompress demo_bag`
- `roslaunch filter the_contents_of the_bag`

* Introspection and Command Tools*

**rosmg**
Displays Message/Service (msg/srv) data structure definitions.

Commands:
- `rosmg show`
- `rosmg list`
- `rosmg info`
- `rosmg package`
- `rosmg packages`

Examples:
- `rosmg show [msg/srv]`
- `rosmg list [msg/srv]`
- `rosmg info [msg/srv]`
- `rosmg package [msg/srv]`
- `rosmg packages [msg/srv]`

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

Commands:
- `rosmode ping`
- `rosmode list`
- `rosmode info`
- `rosmode machine`
- `rosmode kill`

Examples:
- `rosmode ping`
- `rosmode list`
- `rosmode info`
- `rosmode machine`
- `rosmode kill`

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

Commands:
- `rostopic ls`
- `rostopic echo`
- `rostopic info`
- `rostopic list`
- `rostopic pub`
- `rostopic type`

Examples:
- `rostopic info [topic]` Display messages that match a given Python expression.
- `rostopic echo [topic]` Pipe the output of rostopic to rossave to view the msg type.
- `rostopic type [topic]` Pipe the output of rostopic to rossave to view the msg type.

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

Commands:
- `rostopam get [param]`
- `rostopam load [file]`
- `rostopam dump [file]`
- `rostopam delete [file]`

Examples:
- `rostopam get [param]`
- `rostopam load [file]`
- `rostopam dump [file]`
- `rostopam delete [file]`

* Palouse RoboSub Technical Documentation - https://robosub.eecs.wsu.edu/wiki/*
ROS Indigo Cheatsheet

Logging Tools

rqt_console
A tool to display and filter messages published on a topic.
Usage:
$ rqt-console

rqt_bag
A tool for stimulating, inspecting, and replaying log files.
Usage, viewing:
$ rqt_bag log.bag
Usage, bagging:
$ rqt_bag -e press the big red record button.

rqt_logger_level
Change the logger level of ROS nodes. This will increase or decrease the information they log to the screen and rosout.log
Usage:
$ rqt_logger_level

Logging & 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 msgs, srvs, and actions.
Usage:
$ rqt

Intropection & Command Tools

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 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:
$ rosrer tf tf_echo <source_frame> <target_frame>

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

view_frames
A tool for visualizing the full tree of coordinate transforms.
Usage:
$ rosrer tf tools view_frames.py
$ wince frames.png

rqt_plot
A tool for plotting data from ROS topic fields.

Examples:
To graph the data in different plots:
$ rqt_plot /topic1/data1 /topic2/data2
To graph the data all on the same plot:
$ rqt_plot /topic1/data1 /topic2/data2
To graph multiple fields of a message:
$ rqt_plot /topic1/data1/data2/data3

rqt_image_view
A tool for displaying image topics.

Usage:
$ rqt_image_view
ROS Indigo Catkin Workspaces

Create a catkin workspace

Setup and use a new catkin workspace from scratch.

Examples:

```
$ source /opt/ros/hydro/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`.

Examples:

```
$ cd ~/catkin_ws/src
$ wstool init
$ wstool met tutorials --git git://github.com/ros/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]
```

Examples:

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg tutorials std_msgs rosspy 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
```