ROS Indigo Cheatsheet

Filesystem Management Tools

- `rosun` - A tool for inspecting packages.
- `rosun profile` - Fine file and package builds.
- `rosun` - Change directory to a package.
- `rosun` - Package equivalent for ROS.
- `rosun` - List package or stack information.
- `rosun` - Open ROS file in a text editor.
- `rosun copy` - Copy a file from one place to another.
- `rosun install` - Install package system dependencies.
- `rosun` - Displays errors and warnings about a running ROS system or launch file.
- `rosun install` - Create a new ROS stack.
- `rosun exec` - Manage many ros in a workspace.
- `rosun` - Builds a ROS creation workspace.
- `rosun` - Displays package structure and dependences.

Usage:
- `rosun find [package]`
- `rosun [package/subdir]`
- `rosun list [package]`
- `rosun show [package]`
- `rosun install [package]`
- `rosun exec [package]`
- `rosun create [package]`
- `rosun clean [package]`

Start-up and Process Launch Tools

- `roscat` - Starts a roscat package executable with minimal typing.
- `roscat exec` - Starts a roscat package executable command.
- `roscat` - Starts a roscat package executable.
- `roscat` - Launch on a different port.
- `roscat -p 1234 package_file_name.launch` - Launch on the local node.
- `roscat --local package_file_name.launch`

Logging Tools

- `roslaunch` - A tool for registering and playing back of ROS topics.
  - `roslaunch record` - Record a bag file with specified topics.
  - `roslaunch play` - Play content of one or more bag files.
  - `roslaunch compress` - Compress one or more bag files.
  - `roslaunch decompress` - Decompress one or more bag files.
  - `roslaunch filter` - Filter the contents of the bag.

Examples:
- `roslaunch record topic1 topic2`
- `roslaunch play -d demo.bag`
- `roslaunch decompress demo1.bag`
- `roslaunch filter demo2.bag`

Introspection and Command Tools

- `rosmf` - Displays Message/Service (msg/srv) data structure definitions.
  - `rosmf show` - Display the fields in the msg/srv.
  - `rosmf list` - Display a list of all msg/srv.
  - `rosmf find` - Display a list of all msg/srv.
  - `rosmf package` - List all the packages containing the msg/srv.

Examples:
- `rosmf show msg`
- `rosmf show Pose msg`
- `rosmf list [package]`
- `rosmf list sensor_msgs/msg/CameraInfo`
- `rosmf list sensor_msgs/msg/package`

- `rosmake` - Displays debugging information about ROS nodes, including publications, subscriptions and connections.
  - `rosmake ping` - Test connectivity to node.
  - `rosmake list` - List active nodes.
  - `rosmake info` - Print information about a node.
  - `rosmake machine` - List nodes running on a machine.
  - `rosmake kill` - Kill a running node.

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

- `roscfg` - A tool for listing and querying ROS services.
  - `roscfg get` - Get a parameter.
  - `roscfg load` - Load parameters from a file.
  - `roscfg delete` - Delete a parameter.

Examples:
- `roscfg get [parameter]`
- `roscfg load [file]`
- `roscfg delete [parameter]`

- `rostopic` - A tool for displaying information about ROS topics, including publishers, subscribers, publishing rate, and messages.
  - `rostopic LS` - Display all active topics.
  - `rostopic LS` - Print topics by node.
  - `rostopic LS node` - Print information about a specific node.
  - `rostopic LS topic` - Print information about an active topic.

Examples:
- `rostopic LS`
- `rostopic LS node node_name`
- `rostopic LS topic topic_name`

- `rosnode` - A tool for listing and querying ROS services.
  - `rosnode list` - List all the parameters in a namespace.
  - `rosnode list [namespace]`
  - `rosnode list [namespace]`

Examples:
- `rosnode list [namespace]`
- `rosnode list [namespace]`
- `rosnode list [namespace]`

- `rosrun` - A tool for listing and querying ROS services.
  - `rosrun service` - Call the service with the given args.
  - `rosrun service` - Print the service type.
  - `rosrun service` - Print the service type.

Examples:
- `rosrun service [service] [args]`
- `rosrun service [service] [args]`
- `rosrun service [service] [args]`

- `roscommon` - A tool for listing and querying ROS services.
  - `roscommon list` - Print name of provider of service.
  - `roscommon list` - Call the service with the given args.

Examples:
- `roscommon list [service] [args]`
- `roscommon list [service] [args]`
- `roscommon list [service] [args]`

- `roscommon` - A tool for listing and querying ROS services.
  - `roscommon list` - List the arguments of a service.
  - `roscommon list` - Print the service type.

Examples:
- `roscommon list [service] [args]`
- `roscommon list [service] [args]`
- `roscommon list [service] [args]`

- `roscommon` - A tool for listing and querying ROS services.
  - `roscommon list` - Find services by service type.

Examples:
- `roscommon list [service] [args]`
- `roscommon list [service] [args]`
- `roscommon list [service] [args]`
ROS Indigo Cheatsheet

Logging Tools

**ros.log**
A tool for displaying and filtering messages published on specific topics.

Usage:
```bash
$ ros.log
```

**ros.log**
A tool for viewing, logging, and replaying log files.

Usage, viewing:
```bash
$ ros.log [file.log]
```
Usage, logging:
```bash
$ ros.log -p big red log button
```

**ros_level**
Change the logger level of ROS nodes. This will increase or decrease the information they log to the screen and replays.

Usage:
```bash
$ ros.level
```

**ros_logger_level**
Change the logger level of ROS nodes. This will increase or decrease the information they log to the screen and replays.

Usage:
```bash
$ ros_logger_level
```

Introspection & Command Tools

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

Usage:
```bash
$ ros_topic
```

**ros_new, ros_srv, and ros_action**
A tool for viewing available maps, srvs, and actions.

Usage:
```bash
$ ros
```

Usage:
```bash
$ ros_new
```

Usage:
```bash
$ ros_srv
```

Usage:
```bash
$ ros_action
```

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

Usage:
```bash
$ ros_reconfigure
```

Development Environments

**ros_node, and ros_py_con**
Two tools for accessing an os shell and python console respectively.

Usage:
```bash
$ ros
```

Usage:
```bash
$ ros_node
```

**ros_py_con**
A tool for displaying image topics.

Usage:
```bash
$ ros_py_con
```

Data Visualization Tools

**tf**
A tool that prints the information about a particular transformation between a source.frame and a target.frame.
Usage:
```bash
$ ros tf tf.print [source.frame] [target.frame]
```

Examples:
To echo a transform between `/map` and `/odom`:
```bash
$ ros tf echo /map /odom
```

**view_frames**
A tool for visualizing the full tree of coordinate transforms.
Usage:
```bash
$ ros tf,tools view_frames.py
```

**tf**
A tool for visualizing data from ROS topic fields.

Usage:
```bash
$ ros tf
```

Examples:
To graph the data in different plots:
```bash
$ ros tf.plot [topic1] [topic2] [field1]
```
To graph the data all on the same plot:
```bash
$ ros tf.plot [topic1] [topic2] [field1]
```
To graph multiple fields of a message:
```bash
$ ros tf.plot [topic1] [field1] [field2]
```

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

Usage:
```bash
$ ros_reconfigure
```

**tf_image_view**
A tool for displaying image topics.

Usage:
```bash
$ ros_image.view
```
ROS Indigo Catkin Workspaces

Create a catkin workspace

Setup and use a new catkin workspace from scratch.

Examples:
```bash
# 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`.

Examples:
```bash
# cd ~/catkin_ws/src
# wstool init
# wstool get tutorials --git https://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`.

Example:
```bash
# cd ~/catkin_ws/src
# catkin_create_pkg <package-name> [depend1] [depend2]
```

Build all packages in a workspace

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

Examples:
```bash
# cd ~/catkin_ws
# catkin_make
```

Source devel/setup.bash

---

Copyright © 2015 Open Source Robotics Foundation

Copyright © 2015 Willow Garage

---

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