

ROS Cheat Sheet

Filesystem Command-line Tools

rospack/rostack	A tool inspecting packages/stacks .
roscd	Changes directories to a package or stack.
rosls	Lists package or stack information.
roscrcat	Creates a new ROS package.
roscrcat-stack	Creates a new ROS stack.
roscd	Installs ROS package system dependencies.
rosmake	Builds a ROS package.
roswtf	Displays a errors and warnings about a running ROS system or launch file.
rxdeps	Displays package structure and dependencies.

Usage:

```
$ rospack find [package]
$ roscd [package[/subdir]]
$ rosls [package[/subdir]]
$ roscrcat [package_name]
$ rosmake [package]
$ roscd install [package]
$ roswtf or roswtf [file]
$ rxdeps [options]
```

Common Command-line Tools

roscore

A collection of [nodes](#) and programs that are pre-requisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate.

roscore is currently defined as:

```
master
parameter server
rosout
```

Usage:

```
$ roscore
```

rosmmsg/rossrv

rosmmsg/rossrv displays Message/Service (msg/srv) data structure definitions.

Commands:	
rosmmsg show	Display the fields in the msg.
rosmmsg users	Search for code using the msg.
rosmmsg md5	Display the msg md5 sum.
rosmmsg package	List all the messages in a package.
roscd packages	List all the packages with messages.

Examples:

```
Display the Pose msg:
$ rosmmsg show Pose
List the messages in nav_msgs:
$ rosmmsg package nav_msgs
List the files using sensor_msgs/CameraInfo:
$ rosmmsg users sensor_msgs/CameraInfo
```

roscd

roscd allows you to run an executable in an arbitrary package without having to cd (or roscd) there first.

Usage:

```
$ roscd package executable
```

Example:

```
Run turtlesim:
$ roscd turtlesim turtlesim_node
```

roscd

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

Commands:	
roscd ping	Test connectivity to node.
roscd list	List active nodes.
roscd info	Print information about a node.
roscd machine	List nodes running on a particular machine.
roscd kill	Kills a running node.

Examples:

```
Kill all nodes:
$ roscd kill -a
List nodes on a machine:
$ roscd machine aqy.local
Ping all nodes:
$ roscd ping --all
```

roslaunch

Starts ROS nodes locally and remotely via SSH, as well as setting parameters on the parameter server.

Examples:

```
Launch on a different port:
$ roslaunch -p 1234 package filename.launch
Launch a file in a package:
$ roslaunch package filename.launch
Launch on the local nodes:
$ roslaunch --local package filename.launch
```

rostopic

A tool for displaying debug information about ROS [topics](#), including publishers, subscribers, publishing rate, and messages.

Commands:	
rostopic bw	Display bandwidth used by topic.
rostopic echo	Print messages to screen.
rostopic hz	Display publishing rate of topic.
rostopic list	Print information about active topics.
rostopic pub	Publish data to topic.
rostopic type	Print topic type.
rostopic find	Find topics by type.

Examples:

```
Publish hello at 10 Hz:
$ rostopic pub -r 10 /topic_name std_msgs/String hello
Clear the screen after each message is published:
$ rostopic echo -c /topic_name
Display messages that match a given Python expression:
$ rostopic echo --filter "m.data=='foo'" /topic_name
Pipe the output of rostopic to rosmmsg to view the msg type:
$ rostopic type /topic_name | rosmmsg show
```

roscd

A tool for getting and setting ROS [parameters](#) on the parameter server using YAML-encoded files.

Commands:	
roscd set	Set a parameter.
roscd get	Get a parameter.
roscd load	Load parameters from a file.
roscd dump	Dump parameters to a file.
roscd delete	Delete a parameter.
roscd list	List parameter names.

Examples:

```
List all the parameters in a namespace:
$ roscd list /namespace
Setting a list with one as a string, integer, and float:
$ roscd set /foo ["'1'", 1, 1.0]"
Dump only the parameters in a specific namespace to file:
$ roscd dump dump.yaml /namespace
```

rosservice

A tool for listing and querying ROS services.

Commands:	
rosservice list	Print information about active services.
rosservice node	Print the name of the node providing a service.
rosservice call	Call the service with the given args.
rosservice args	List the arguments of a service.
rosservice type	Print the service type.
rosservice uri	Print the service ROSRPC uri.
rosservice find	Find services by service type.

Examples:

```
Call a service from the command-line:
$ rosservice call /add_two_ints 1 2
Pipe the output of rosservice to rossrv to view the srv type:
$ rosservice type add_two_ints | rossrv show
Display all services of a particular type:
$ rosservice find rospy_tutorials/AddTwoInts
```

Logging Command-line Tools

roscap

This is a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoids deserialization and reserialization of the messages.

roscap record will generate a “.bag” file (so named for historical reasons) with the contents of all topics that you pass to it.

Examples:

Record all topics:

```
$ roscap record -a
```

Record select topics:

```
$ roscap record topic1 topic2
```

roscap play will take the contents of one or more bag file, and play them back in a time-synchronized fashion.

Examples:

Replay all messages without waiting:

```
$ roscap play -a demo_log.bag
```

Replay several bag files at once:

```
$ roscap play demo1.bag demo2.bag
```

Graphical Tools

rxgraph

Displays a graph of the ROS nodes that are currently running, as well as the ROS topics that connect them.

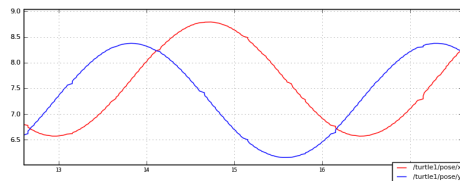


Usage:

```
$ rxgraph
```

rxplot

A tool for plotting data from one or more ROS topic fields using matplotlib.



Examples:

To graph the data in different plots:

```
$ rxplot /topic1/field1 /topic2/field2
```

To graph the data all on the same plot:

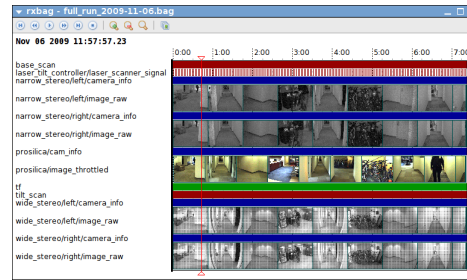
```
$ rxplot /topic1/field1,/topic2/field2
```

To graph multiple fields of a message:

```
$ rxplot /topic1/field1:field2:field3
```

rxbag

A tool for visualizing, inspecting, and replaying histories (bag files) of ROS messages.

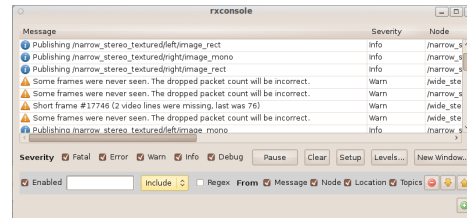


Usage:

```
$ rxbag bag_file.bag
```

rxconsole

A tool for displaying and filtering messages published on roscout.



Usage:

```
$ rxconsole
```

tf Command-line Tools

tf_echo

A tool that prints the information about a particular transformation between a source_frame and a target_frame.

Usage:

```
$ roscout tf tf_echo <source_frame> <target_frame>
```

Examples:

To echo the transform between /map and /odom:

```
$ roscout tf tf_echo /map /odom
```

view_frames

A tool for visualizing the full tree of coordinate transforms.

Usage:

```
$ roscout tf view_frames
```

```
$ evince frames.pdf
```