

ROS Cheat Sheet

Filesystem Command-line Tools

rospack /rosstack	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.
roscrcat	Installs ROS package system dependencies.
roscrcat	Builds a ROS package.
roscrcat	Displays a errors and warnings about a running ROS system or launch file.
roscrcat	Displays package structure and dependencies.

Usage:

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

Common Command-line Tools

roscrcat

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

roscrcat is currently defined as:

```
master
parameter server
rosout
```

Usage:

```
$ roscrcat
```

roscrcat/roscrcat

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

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

Examples:

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

roscrcat

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

Usage:

```
$ roscrcat package executable
```

Example:

```
Run turtlesim:
$ roscrcat turtlesim turtlesim_node
```

roscrcat

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

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

Examples:

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

roscrcat

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

Examples:

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

roscrcat

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

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

Examples:

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

roscrcat

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

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

Examples:

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

roscrcat

A tool for listing and querying ROS services.

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

Examples:

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

Logging Command-line Tools

rosvbag

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.

rosvbag 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:

```
$ rosvbag record -a
```

Record select topics:

```
$ rosvbag record topic1 topic2
```

rosvbag 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:

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

Replay several bag files at once:

```
$ rosvbag 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.

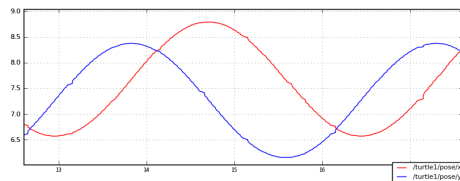


Usage:

```
$ 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:

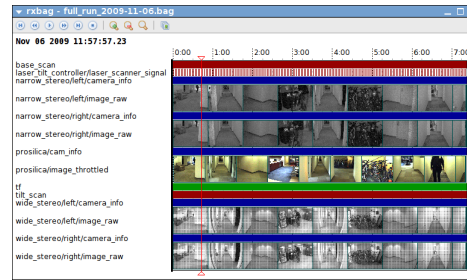
```
$ rxplot /topic1/field1,/topic2/field2
```

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.

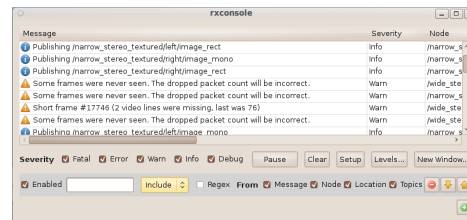


Usage:

```
$ rxbag bag_file.bag
```

rxconsole

A tool for displaying and filtering messages published on rosvout.



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:

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

Examples:

To echo the transform between /map and /odom:

```
$ rosvrun tf tf_echo /map /odom
```

view_frames

A tool for visualizing the full tree of coordinate transforms.

Usage:

```
$ rosvrun tf view_frames
```

```
$ evince frames.pdf
```