Dynamic Reconfigure

⇒ This package provides a means to change node parameters at any time without having to restart the node.

⇒ Client program e.g. GUI. can query the node for the set of reconfigurable Parameters, including their names, types and range, and present a customized interface to the user.

* reconfigure_gui

⇒ The reconfigure_gui tool is now provided in rqt.

↳ rqt_reconfigure

* dynparam Command-line tool

rosrun dynamic_reconfigure dynparam COMMAND

⇒ The Currently Supported commands are:

① dynparam list

↳ list all the reconfigurable nodes in alphabetic order.

② dynparam get node_name

↳ Get the Configuration of a reconfigurable node.

-t secs

↳ Timeout in seconds.

③ dynparam set

↳ set node_name parameter_name parameter_value

↳ set node_name yaml_dictionary

-t secs

↳ Timeout in Seconds.

④ dynparam set_from_parameters

↳ set_from_parameter node_name

-t secs

↳ Timeout in Seconds

⑤ dynparam dump node_name file.yaml

⇒ Dump the Configuration of a reconfigurable node to a file.

-t secs.

② dynparam load nodename file.yaml

↳ Load Configuration for a node from a file.

-t secs.

Dynamic Reconfigure Tutorial

1

How to write your first .cfg File

⇒ In your package create a cfg directory, this is where all cfg files lives:

Tutorial ~~Example~~.cfg file

```
#!/usr/bin/env python
PACKAGE = "dynamic_tutorials"

from dynamic_reconfigure.parameter_generator_catkin import *

gen = ParameterGenerator()

gen.add("int_param", int_t, 0, "An Integer Param"
        , 50, 0, 100)

exit(gen.generate(PACKAGE, "dynamic_tutorials", ~~"Example"~~ "Tutorial"))
```

⇒ An exit:

- The second parameter is the name of the node that could run it
  (used to generate documentation only)
- Third parameter is a name prefix for the generated file
  ("<name>Config.h" for C++)

Note: The third parameter should be equal to the cfg file name, without extension.

⇒ In order to make the cfg file useable, it must be executable:

chmod a+x cfg/Tutorial.cfg

⇒ We also need to edit the CMakeLists.txt file