Contents

1. Names

1.1 Graph Resource Names

Graph Resource Names provide a hierarchical naming structure that is used for all resources in a ROS Computation Graph, such as Nodes (/Nodes), Parameters (/Parameter%20Server), Topics (/Topics), and Services (/Services). These names are very powerful in ROS and central to how larger and more complicated systems are composed in ROS, so it is critical to understand how these names work and how you can manipulate them.

Before we describe names further, here are some example names:

- `/` (the global namespace)
- `/foo`
- `/stanford/robot/name`
- `/wg/node1`

Graph Resource Names are an important mechanism in ROS for providing encapsulation. Each resource is defined within a namespace, which it may share with many other resources. In general, resources can *create* resources within their namespace and they can *access* resources within or above their own namespace. Connections can be made between resources in distinct namespaces, but this is generally done by integration code above both namespaces. This encapsulation isolates different portions of the system from accidentally grabbing the wrong named resource or globally hijacking names.

Names are resolved relatively, so resources do not need to be aware of which namespace they are in. This simplifies programming as nodes that work together can be written as if they are all in the top-level namespace. When these Nodes are integrated into a larger system, they can be *pushed down* into a namespace that defines their collection of code. For example, one could take a Stanford demo and a Willow Garage demo and merge them into a new demo with `stanford` and `wg` subgraphs. If both demos had a Node named 'camera', they would not conflict. Tools (e.g. graph visualization) as well as parameters (e.g. `demo_name`) that need to be visible to the entire graph can be created by top-level Nodes.

1.1.1 Valid Names

A valid name has the following characteristics:

1. First character is an alpha character ([a-z|A-Z]), tilde (~) or forward slash (`/`)

2. Subsequent characters can be alphanumeric ([0-9|a-z|A-Z]), underscores (_), or forward slashes (/)

Exception: base names (described below) cannot have forward slashes (/) or tildes (~) in them.

1.1.2 Resolving

There are four types of Graph Resource Names in ROS: *base*, *relative*, *global*, and *private*, which have the following syntax:

- `base`
- `relative/name`
- `/global/name`
- `~private/name`

By default, resolution is done *relative* to the node's namespace. For example, the node `/wg/node1` has the namespace `/wg`, so the name `node2` will resolve to `/wg/node2`.

Names with no namespace qualifiers whatsoever are *base* names. Base names are actually a subclass of relative names and have the same resolution rules. Base names are most frequently used to initialize the node name.

Names that start with a "/" are *global* -- they are considered fully resolved. Global names should be avoided as much as possible as they limit code portability.

Names that start with a "~" are *private*. They convert the node's name into a namespace. For example, `node1` in namespace `/wg/` has the private namespace `/wg/node1`. Private names are useful for passing parameters to a specific node via the parameter server.

Here are some name resolution examples:

Node	**Relative** (default)	**Global**	**Private**
`/node1`	`bar` -> `/bar`	`/bar` -> `/bar`	`~bar` -> `/node1/bar`
`/wg/node2`	`bar` -> `/wg/bar`	`/bar` -> `/bar`	`~bar` -> `/wg/node2/bar`
`/wg/node3`	`foo/bar` -> `/wg/foo/bar`	`/foo/bar` -> `/foo/bar`	`~foo/bar` -> `/wg/node3/foo/bar`

1.1.3 Remapping

Any name within a ROS Node can be remapped when the Node is launched at the command-line. For more information on this feature, see Remapping Arguments (/Remapping%20Arguments).

1.2 Package Resource Names

Package Resource Names are used in ROS with Filesystem-Level concepts to simplify the process of referring to files and data types on disk. Package Resource Names are very simple: they are just the name of the Package (/Packages) that the resource is in plus the name of the resource. For example, the name "std_msgs/String" refers to the "String" message type in the "std_msgs" Package.

Some of the ROS-related files that may be referred to using Package Resource Names include:

- Message (msg) types (/msg)
- Service (srv) types (/srv)

- Node types (/Nodes)

Package Resource Names are very similar to file paths, except they are much shorter. This is due to the ability of ROS to locate Packages on disk and make additional assumptions about their contents. For example, Message descriptions are always stored in the `msg` subdirectory and have the `.msg` extension, so `std_msgs/String` is shorthand for `path/to/std_msgs/msg/String.msg`. Similarly, the Node type `foo/bar` is equivalent to searching for a file named `bar` in Package `foo` with executable permissions.

1.2.1 Valid Names

Package Resource Names have strict naming rules as they are often used in auto-generated code. For this reason, a ROS package (/Packages) cannot have special characters other than an underscore, and they must start with an alphabetical character. A valid name has the following characteristics:

1. First character is an alpha character ([a-z|A-Z])
2. Subsequent characters can be alphanumeric ([0-9|a-z|A-Z]), underscores (_) or a forward slash (/)
3. There is at most one forward slash ('/').

1.3 Code API

roscpp::names API reference (http://docs.ros.org/indigo/api/roscpp/html/namespaceros_1_1names.html) (ROS Indigo)

Wiki: Names (last edited 2015-07-15 14:08:27 by AdamAllevato (/AdamAllevato))