

Robotics: Principles and Practice

Module 2: The Robot Operating System (ROS)

Lecture 4: Writing ROS software in C++: services

David Vernon
Carnegie Mellon University Africa

www.vernon.eu

Writing ROS Software

- Creating a ROS workspace and a ROS package
- Writing ROS programs

1. Example program: Hello World!

2. Example program to publish messages

Send velocity messages on `/turtle1/cmd_vel`

3. Example program to subscribe to messages

Receive pose messages on `/turtle1/pose`

Writing ROS Software

- Writing ROS programs

4. Example program to use services

This lecture

`/reset`

`/clear`

`/turtle1/set_pen`

`/turtle1/teleport_absolute`

Example Client Program to Use Services

`/clear, /turtle1/set_pen, /turtle1/teleport_absolute`

2.1.3 Services

`clear (std_srvs/Empty)`

Clears the turtlesim background and sets the color to the value of the background parameters.

`reset (std_srvs/Empty)`

Resets the turtlesim to the start configuration and sets the background color to the value of the background.

`kill (turtlesim/Kill)`

Kills a turtle by name.

`spawn (turtlesim/Spawn)`

Spawns a turtle at (x, y, theta) and returns the name of the turtle. Also will take name for argument but will fail if a duplicate name.

`turtleX/set_pen (turtlesim/SetPen)`

Sets the pen's color (r g b), width (width), and turns the pen on and off (off).

`turtleX/teleport_absolute (turtlesim/TeleportAbsolute)`

Teleports the turtleX to (x, y, theta).

`turtleX/teleport_relative (turtlesim/TeleportRelative)`

Teleports the turtleX a linear and angular distance from the turtles current position.

<http://wiki.ros.org/turtlesim>

Example Client Program to Use Services

`/clear, /turtle1/set_pen, /turtle1/teleport_absolute`

Make sure you are in the agitr sub-directory

```
~/workspace/ros/src$ cd ~/workspace/ros/src/agitr
```


Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

Move to the **agitr/src** sub-directory

```
~/workspace/ros/src/agitr$ cd src
```

```
~/workspace/ros/src/agitr/src$
```

Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

Edit `useservices.cpp` and insert the following code

```
/* This client program uses a sample of turtlesim services */
/* /clear, /turtle1/set_pen, and /turtle1/telepor_absolute */

#include <ros/ros.h>
#include <turtlesim/TeleportAbsolute.h> // for turtle1/teleport_absolute service
#include <turtlesim/SetPen.h>          // for turtle1/set_pen service
#include <std_srvs/Empty.h>           // for reset and clear services

int main(int argc, char **argv) {

    bool success = true;

    /* Initialize the ROS system and become a node */
    ros::init(argc, argv, "mystudentid"); // Initialize the ROS system
    ros::NodeHandle nh;                    // Become a node

    /* Create client objects for the required services */
    ros::service::waitForService("turtle1/teleport_absolute");
    ros::ServiceClient teleportClient = nh.serviceClient<turtlesim::TeleportAbsolute>("turtle1/teleport_absolute");

    ros::service::waitForService("turtle1/set_pen");
    ros::ServiceClient setpenClient = nh.serviceClient<turtlesim::SetPen>("turtle1/set_pen");

    ros::service::waitForService("clear");
    ros::ServiceClient clearClient = nh.serviceClient<std_srvs::Empty>("clear");
```


Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

```
/* Create the service objects services */
turtlesim::TeleportAbsolute teleport_arguments; // reposition the turtle without locomotion
turtlesim::SetPen          pen_arguments;      // turn the pen on/off and change colour
std_srvs::Empty            clear_arguments;    // clear the background

/* clear the simulator background */
success = clearClient.call(clear_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to clear" );
}

/* turn the pen off so that we don't see a trace when the turtle teleports */
pen_arguments.request.off = 1;
success = setpenClient.call(pen_arguments);
if (!success) {
    ROS_ERROR_STREAM("TurtlePen failed to switch off");
}

teleport_arguments.request.x      = 2.5;      // location
teleport_arguments.request.y      = 3.5;      // coordinates
teleport_arguments.request.theta = 3.14159 / 2; // facing up, i.e. 90 degrees

success = teleportClient.call(teleport_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to teleport" );
}
```

Example Client Program to Use Services

`/clear, /turtle1/set_pen, /turtle1/teleport_absolute`

```
/* turn the pen again so that we do see a trace when the turtle moves later on */
pen_arguments.request.off    = 0;
pen_arguments.request.r     = 255; // white
pen_arguments.request.g     = 255; // pen
pen_arguments.request.b     = 255; // colour
pen_arguments.request.width = 1;  // narrow line

success = setpenClient.call(pen_arguments);
if (!success) {
    ROS_ERROR_STREAM("TurtlePen failed to switch on");
}
}
```

Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

Edit `useservices.cpp` and insert the following code

```
/* This client program uses a sample of turtlesim services */
/* /clear, /turtle1/set_pen, and /turtle1/telepor_absolute */

#include <ros/ros.h>
#include <turtlesim/TeleportAbsolute.h> // for turtle1/teleport_absolute service
#include <turtlesim/SetPen.h>          // for turtle1/set_pen service
#include <std_srvs/Empty.h>           // for reset and clear services

int main(int argc, char **argv) {

    bool success = true;

    /* Initialize the ROS system and become a node */
    ros::init(argc, argv, "mystudentid"); // Initialize the ROS system
    ros::NodeHandle nh;                    // Become a node

    /* Create client objects for the required services */
    ros::service::waitForService("turtle1/teleport_absolute");
    ros::ServiceClient teleportClient = nh.serviceClient<turtlesim::TeleportAbsolute>("turtle1/teleport_absolute");

    ros::service::waitForService("turtle1/set_pen");
    ros::ServiceClient setpenClient = nh.serviceClient<turtlesim::SetPen>("turtle1/set_pen");

    ros::service::waitForService("clear");
    ros::ServiceClient clearClient = nh.serviceClient<std_srvs::Empty>("clear");
```

`waitForService()` waits for a service to be advertised and available. Block until it is.
This is defensive programming; normally it works without it.

Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

Edit `useservices.cpp` and insert the following code

```
/* This client program uses a sample of turtlesim services */
/* /clear, /turtle1/set_pen, and /turtle1/telepor_absolute */

#include <ros/ros.h>
#include <turtlesim/TeleportAbsolute.h> // for turtle1/teleport_absolute service
#include <turtlesim/SetPen.h>          // for turtle1/set_pen service
#include <std_srvs/Empty.h>           // for reset and clear services

int main(int argc, char **argv) {

    bool success = true;

    /* Initialize the ROS system and become a node */
    ros::init(argc, argv, "mystudentid"); // Initialize the ROS system
    ros::NodeHandle nh;                    // Become a node

    /* Create client objects for the required services */
    ros::service::waitForService("turtle1/teleport_absolute");
    ros::ServiceClient teleportClient = nh.serviceClient<turtlesim::TeleportAbsolute>("turtle1/teleport_absolute");

    ros::service::waitForService("turtle1/set_pen");
    ros::ServiceClient setpenClient = nh.serviceClient<turtlesim::SetPen>("turtle1/set_pen");

    ros::service::waitForService("clear");
    ros::ServiceClient clearClient = nh.serviceClient<std_srvs::Empty>("clear");
```

You need to identify the service type when creating the client object



Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

Edit `useservices.cpp` and insert the following code

```
/* This client program uses a sample of turtlesim services */
/* /clear, /turtle1/set_pen, and /turtle1/telepor_absolute */

#include <ros/ros.h>
#include <turtlesim/TeleportAbsolute.h> // for turtle1/teleport_absolute service
#include <turtlesim/SetPen.h> // for turtle1/set_pen service
#include <std_srvs/Empty.h> // for reset and clear services

int main(int argc, char **argv) {

    bool success = true;

    /* Initialize the ROS system and become a node */
    ros::init(argc, argv, "mystudentid"); // Initialize the ROS system
    ros::NodeHandle nh; // Become a node

    /* Create client objects for the required services */
    ros::service::waitForService("turtle1/teleport_absolute");
    ros::ServiceClient teleportClient = nh.serviceClient<turtlesim::TeleportAbsolute>("turtle1/teleport_absolute");

    ros::service::waitForService("turtle1/set_pen");
    ros::ServiceClient setpenClient = nh.serviceClient<turtlesim::SetPen>("turtle1/set_pen");

    ros::service::waitForService("clear");
    ros::ServiceClient clearClient = nh.serviceClient<std_srvs::Empty>("clear");
```

The service types are defined in these include files

Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

Edit `useservices.cpp` and insert the following code

```
/* This client program uses a sample of turtlesim services */
/* /clear, /turtle1/set_pen, and /turtle1/telepor_absolute */

#include <ros/ros.h>
#include <turtlesim/TeleportAbsolute.h> // for turtle1/teleport_absolute service
#include <turtlesim/SetPen.h>          // for turtle1/set_pen service
#include <std_srvs/Empty.h>           // for reset and clear services

int main(int argc, char **argv) {

    bool success = true;

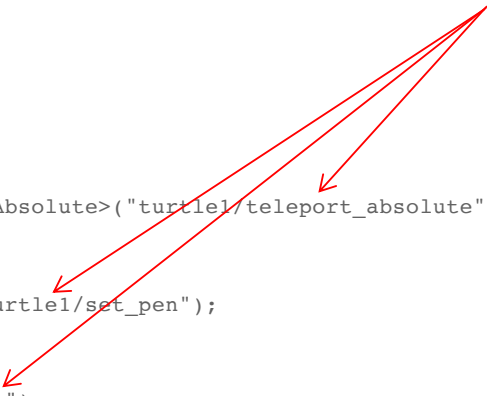
    /* Initialize the ROS system and become a node */
    ros::init(argc, argv, "mystudentid"); // Initialize the ROS system
    ros::NodeHandle nh;                    // Become a node

    /* Create client objects for the required services */
    ros::service::waitForService("turtle1/teleport_absolute");
    ros::ServiceClient teleportClient = nh.serviceClient<turtlesim::TeleportAbsolute>("turtle1/teleport_absolute");

    ros::service::waitForService("turtle1/set_pen");
    ros::ServiceClient setpenClient = nh.serviceClient<turtlesim::SetPen>("turtle1/set_pen");

    ros::service::waitForService("clear");
    ros::ServiceClient clearClient = nh.serviceClient<std_srvs::Empty>("clear");
```

You also need to define a string naming the service you want to call



Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

```
/* Create the service objects services */
turtlesim::TeleportAbsolute teleport_arguments; // reposition the turtle without locomotion
turtlesim::SetPen          pen_arguments;      // turn the pen on/off and change colour
std_srvs::Empty            clear_arguments;    // clear the background

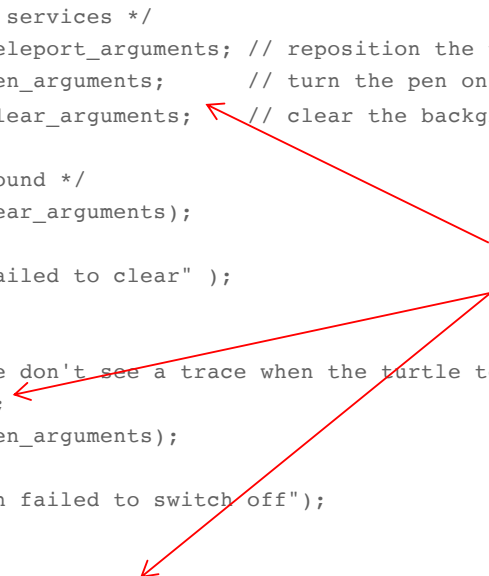
/* clear the simulator background */
success = clearClient.call(clear_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to clear" );
}

/* turn the pen off so that we don't see a trace when the turtle teleports */
pen_arguments.request.off = 0;
success = setpenClient.call(pen_arguments);
if (!success) {
    ROS_ERROR_STREAM("TurtlePen failed to switch off");
}

teleport_arguments.request.x      = 2.5;      // location
teleport_arguments.request.y      = 3.5;      // coordinates
teleport_arguments.request.theta = 3.14159 / 2; // facing up, i.e. 90 degrees

success = teleportClient.call(teleport_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to teleport" );
}
```

We instantiate the service classes as service objects so that we can assign values to the request members



Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

```
/* Create the service objects services */
turtlesim::TeleportAbsolute teleport_arguments; // reposition the turtle without locomotion
turtlesim::SetPen          pen_arguments;      // turn the pen on/off and change colour
std_srvs::Empty            clear_arguments;    // clear the background

/* clear the simulator background */
success = clearClient.call(clear_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to clear" );
}

/* turn the pen off so that we don't see a trace when the turtle teleports */
pen_arguments.request.off = 1; // 0 (false) for on; non-zero (true) for off
success = setpenClient.call(pen_arguments);
if (!success) {
    ROS_ERROR_STREAM("TurtlePen failed to switch off");
}

teleport_arguments.request.x      = 2.5;      // location
teleport_arguments.request.y      = 3.5;      // coordinates
teleport_arguments.request.theta = 3.14159 / 2; // facing up, i.e. 90 degrees

success = teleportClient.call(teleport_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to teleport" );
}
```

We call the service with the associated arguments

Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

```
/* Create the service objects services */
turtlesim::TeleportAbsolute teleport_arguments; // reposition the turtle without locomotion
turtlesim::SetPen          pen_arguments;      // turn the pen on/off and change colour
std_srvs::Empty            clear_arguments;    // clear the background

/* clear the simulator background */
success = clearClient.call(clear_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to clear" );
}

/* turn the pen off so that we don't see a trace when the turtle teleports */
pen_arguments.request.off = 1;
success = setpenClient.call(pen_arguments);
if (!success) {
    ROS_ERROR_STREAM("TurtlePen failed to switch off");
}

teleport_arguments.request.x      = 2.5;      // location
teleport_arguments.request.y      = 3.5;      // coordinates
teleport_arguments.request.theta = 3.14159 / 2; // facing up, i.e. 90 degrees

success = teleportClient.call(teleport_arguments);
if (!success) {
    ROS_ERROR_STREAM("Turtle failed to teleport");
}
```

Check to ensure the service call was successful and report an error if not

Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

```
/* turn the pen again so that we do see a trace when the turtle moves later on */
pen_arguments.request.off    = 0;  // 0 (false) for on; non-zero (true) for off
pen_arguments.request.r     = 255; // white
pen_arguments.request.g     = 255; // pen
pen_arguments.request.b     = 255; // colour
pen_arguments.request.width = 1;  // narrow line

success = setpenClient.call(pen_arguments);
if (!success) {
    ROS_ERROR_STREAM("TurtlePen failed to switch on");
}
}
```

Example Client Program to Use Services

/clear, /turtle1/set_pen, /turtle1/teleport_absolute

Build the workspace to compile the program

- Make sure you are in the workspace directory

```
~/workspace/ros/src/agitr/src$ cd ~/workspace/ros  
~/workspace/rosr$
```

- Run catkin_make

```
~/workspace/ros$ catkin_make
```

Example Client Program to Use Services

`/clear, /turtle1/set_pen, /turtle1/teleport_absolute`

If you have not already done it, open a terminal and enter

```
~$ roscore
```

If you have not already done it, open a second terminal and enter

```
~$ rosrunc turtlesim turtlesim_node
```

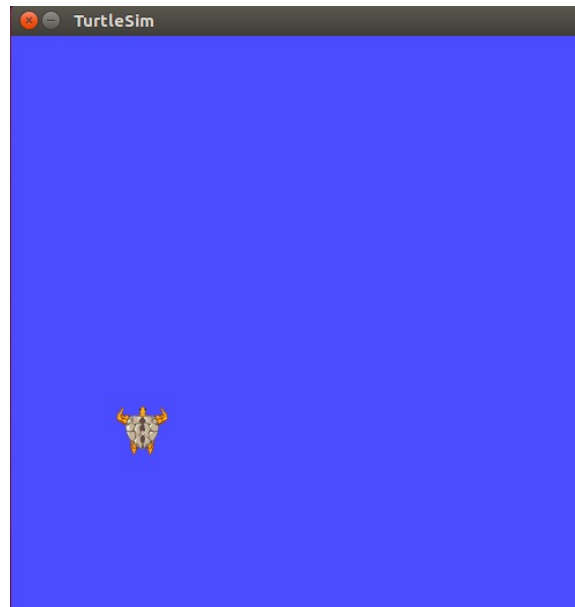
Open a third terminal and enter

```
~$ rosrunc agitr useservices
```

Example Client Program to Use Services

`/clear, /turtle1/set_pen, /turtle1/teleport_absolute`

If everything works correctly, you should see the following

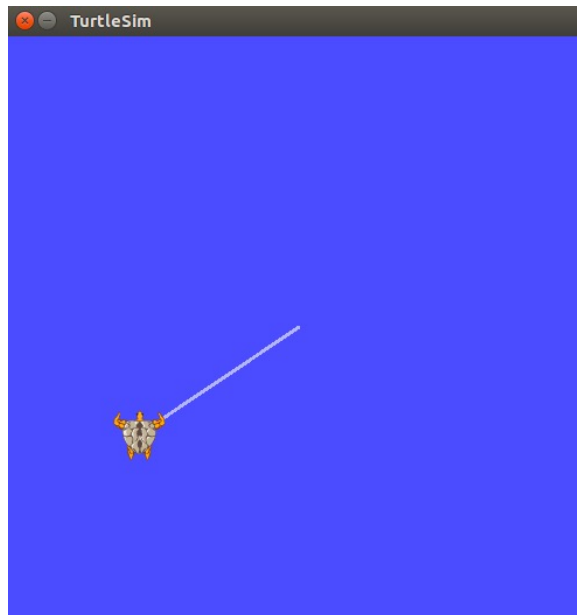


Example Client Program to Use Services

`/clear, /turtle1/set_pen, /turtle1/teleport_absolute`

If you comment out the **first call** to the `set_pen` service to turn the pen off
run `catkin_make`

and run the program again, you should see the following



Example Client Program to Use Services

`/clear, /turtle1/set_pen, /turtle1/teleport_absolute`

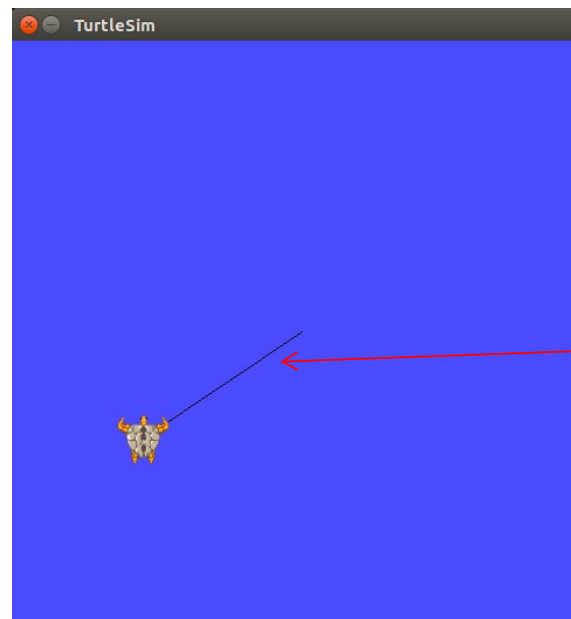
On the other hand, if you either

comment out the statement that assigns the value 1 to `pen_arguments.request.off` (i.e. true)

or

assign the value 0 to `pen_arguments.request.off` (i.e. false),

you should see the following



Why is the line black and thin?

Because when we call the service without assigning values to the pen r, g, and b and width arguments, the service uses default values of zero

whereas when we simply didn't call the service, the values used are those that the turtlesim uses when initializing the simulator, i.e. white thick pen on

ROS Resources

Wiki	http://wiki.ros.org/
Installation	http://wiki.ros.org/ROS/Installation
Tutorials	http://wiki.ros.org/ROS/Tutorials
Tutorial Videos	http://www.youtube.com/playlist?list=PLDC89965A56E6A8D6
ROS Cheat Sheet	http://www.vernon.eu/RPP/ROS_Cheatsheet.pdf

Recommended Reading

http://wiki.ros.org/catkin/Tutorials/create_a_workspace

<http://wiki.ros.org/ROS/Tutorials/CreatingPackage>

<http://wiki.ros.org/roscpp/Overview/InitializationandShutdown>

<http://wiki.ros.org/roscpp/Overview/NodeHandles>

<http://wiki.ros.org/ROS/Tutorials/BuildingPackages>

[http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber\(c++\)](http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(c++))

J. M. O'Kane, A Gentle Introduction to ROS, 2014.

<https://cse.sc.edu/~jokane/agitr/>