### Задания

1. Один спутник посылает другому секретную информацию - координаты секретных объектов. Посылать два числа в открытую слишком рискованно - противник может их перехватить и понять, что ведётся слежка за секретными объектами. Поэтому перед отправкой сообщения необходимо закодировать в строку и передать сообщение только с одной строкой. Второй спутник должен раскодировать сообщение.
2. Разведчик должен передать некоторую информацию своему сообщнику. Договорились встретиться в толпе и передавать информацию в открытом виде, чтобы их разговор слышали все и он не вызвал подозоений. Однако для того, чтобы простой прохожий не понял их речь, они договорились помимо значащих сообщений передавать незначащие. К каждому предложению добавляется число. Если оно чётное - предложение необходимо принять к сведению, если нечётное - отбросить.
3. Бухгалтер раздаёт зарплату сотрудникам довольно специфическим образом: собирает всех сотрудников в одной комнате и выкрикивает фамилию и соответствующую сумму. Одному сотруднику необходимо услышать среди всех выкрикнутых фамилий свою и обрадоваться, если его зарплата оказалась больше, чем у остальных.
4. Ультратоталитарное общество. На работе можно общаться только по поводу работы. В одном офисе работают двое влюблённых, которые хотят договориться о встрече. Общаться они могут только через устную речь. Чтобы фсб не узнало, что они общаются на отвлечённые темы, они договорилиьс, что место и время встречи назначает тот, работа которого связана с постоянным произнесением речей. Информативными для товарища являются сообщения под номерами 5, 10, 15 и так далее.
5. В небесное пространство устремляется боевая ракета, которая летит в определённые координаты. Защищающийся не знает, куда прилетит ракета. Он может в три любые точки на карте поставить противоракеты. Если защищающийся угадал, то с вероятностью 80% нападающая ракета сбивается. Если не угадал, то ракета сбивается с вероятностью 5%.

### Формальные требования

- Необходимо создать три отдельных пакета: для сообщения, для писателя и для читателя
- В каждом пакете должна быть ровно одна нода

### Методические указания

Message.
Сообщение создаётся в иерархии

```
<project_name_folder>
┗━━━src
      ┗━━━<package_folder>
            ┗━━━ msg
                  ┣━━━ <message_file_name #1>.msg
                  ┣━━━ <message_file_name #2>.msg
                  ┗━━━ ...
```

В файле <message_file_name>.msg содержатся поля сообщения разделённые символом переноса строки. Может содержать типы
int8, int16, int32, int64 (плюс uint*)
float32, float64 ээ string (конвертируется в std::string)
time, duration

другие файлы .msg
Массивы_переменной_длинны[] и массивы_фиксированной_длины[C] (конвертируются в std::vector)

Для успешной генерации файла класса-сообщения по файлу прототипа .msg требуется в package.xml добавить

```
<build_depend>message_generation</build_depend>
<run_depend>message_generation</run_depend>
<run_depend>message_runtime</run_depend>
```

Обратите внимание, что для build-depend-а достаточно только message_generation, а для run-depend-а добавляется ещё и message_runtime.

Кроме того в CMakeFile необходимо добавить в

```
find_package (catkin REQUIRED COMPONENTS …)
```

такие компоненты как std_msgs и message_generation.
Также написать

```
add_message_files(
  FILES
  <message_file_name #1>.msg
  <message_file_name #2>.msg
  ...
)
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

При описании publisher-а и subscriber-а необходимо вначале подключить созданный файл сообщения

```
#include “<package_name>/<message_file_name #1>.h”
#include “<package_name>/<message_file_name #2>.h”
…
```

Причём message_file_name должно совпадать с именем файла .msg, который был создан ранее (класс, представляющий это сообщение будет называться также).

Publisher.
В функции main() у publisher-а необходимо создать

```
ros::NodeHandle <node_handle_name>;
```

и сообщить ему о нашем желании передавать сообщения типа <package_name>::<message> (<message> совпадает с <message_file_name>). Пусть в данном примере это будет my_mess из пакета my_package.
Это делается командой

```
ros::Publisher <publisher_name> =
<node_handle_name>.advertise<my_package::my_mess>("<topic_name>",<size>);
```

где <topic_name> это имя топика, через который будут общаться publisher и subscriber; а <size> - размер буфера сообщений (а треугольные скобочки после advertize - это конкретизация шаблонной функции).

Затем создаётся сообщение

```
my_package::my_mess <message_name>;
```

И отправляется в топик:

```
<publisher_name>.publish(<message_name>);
```

Subscriber.
Для subscriber-а необходимо описать функцию-handler, которая будет обрабатывать принятые сообщения:

```
void <function_name>(const my_package::my_mess& <message_name>){
  // body
}
```

А в теле main() необходимо создать подписчика:

```
ros::NodeHandle <node_handle_name>;
ros::Subscriber <subscriber_name> =
   <node_handle_name>.subscribe("<topic_name>", <size>, &<function_name>);
```

Обратите внимание, что в функцию subscribe() передаётся указатель на функцию-обработчик.

В package.xml файле требуется указать depend-ы (build и run) на roscpp и на пакеты, в которых были описаны сообщения, если они создавались в других пакетах. CMakeLists.txt в обоих случаях должны выглядеть следующим образом:

```
cmake_minimum_required(VERSION 2.8.3)
project(<project_name>)
find_package(catkin REQUIRED COMPONENTS roscpp <package_msg>)
catkin_package()
include_directories(include  ${catkin_INCLUDE_DIRS})
add_executable(<exe name> <source_file#1>.cpp <source_file#2>.cpp ...)
target_link_libraries(<exe name> ${catkin_LIBRARIES})
```

From:
https://se.moevm.info/ - **МОЭВМ Вики [se.moevm.info]**

Permanent link:
**https://se.moevm.info/doku.php/courses:ros:lab2**

Last update: