深蓝学院 shenlanxueyuan.com

古月居 GYH.AI

ROS理论与实践

—— 第3讲：ROS通信编程

主讲人 胡春旭

机器人博客“古月居”博主

《ROS机器人开发实践》作者

武汉精锋微控科技有限公司 联合创始人

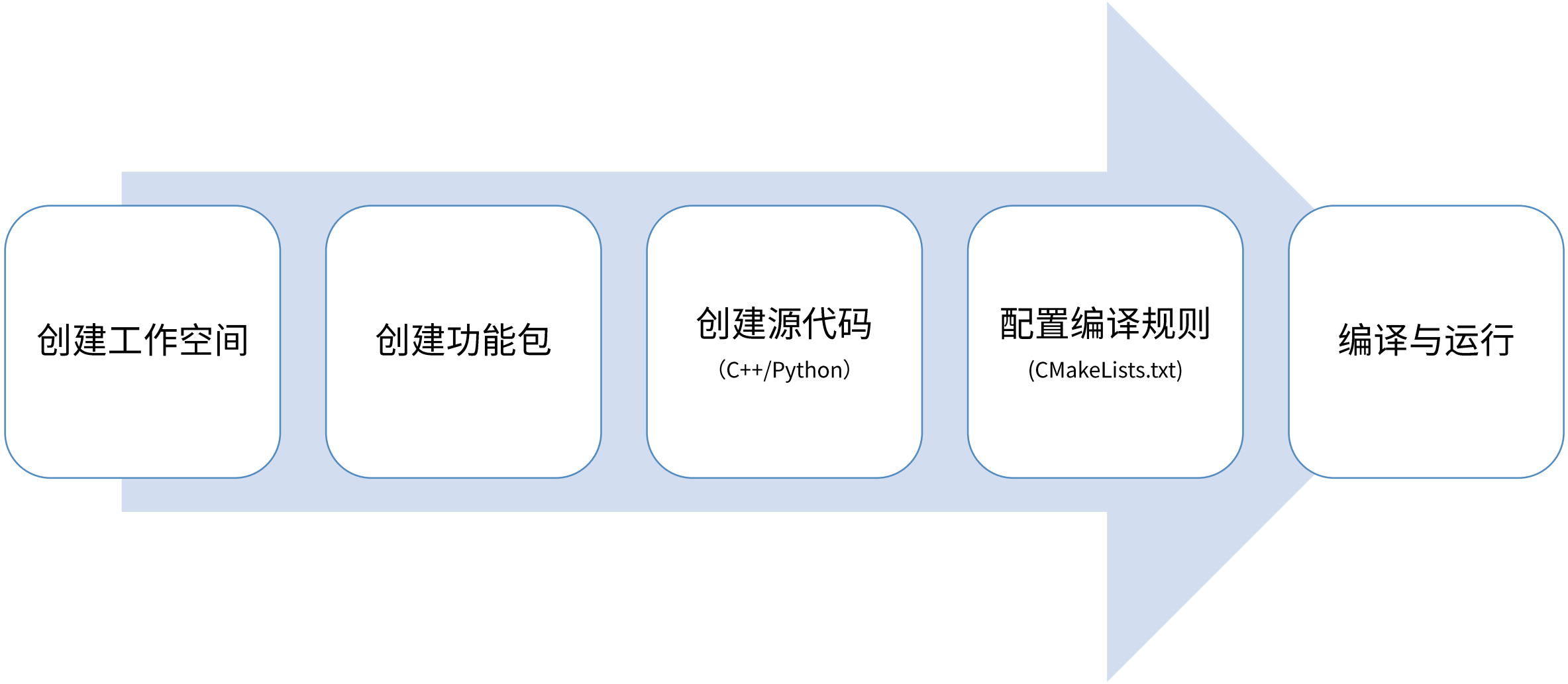
华中科技大学 自动化学院 硕士

目录

古月居 GYH.AI

1. ROS项目开发流程

1. ROS项目开发流程



1. ROS项目开发流程

工作空间（workspace）是一个存放工程开发相关文件的文件夹。

- **src**：代码空间（Source Space）
- **build**：编译空间（Build Space）
- **devel**：开发空间（Development Space）
- **install**：安装空间（Install Space）

```
workspace_folder/          -- WORKSPACE
  src/                     -- SOURCE SPACE
    CMakeLists.txt         -- The 'toplevel' CMake file
    package_1/
      CMakeLists.txt
      package.xml
      ...
    package_n/
      CMakeLists.txt
      package.xml
      ...
  build/                   -- BUILD SPACE
    CATKIN_IGNORE          -- Keeps catkin from walking this directory
  devel/                   -- DEVELOPMENT SPACE (set by CATKIN_DEVEL_PREFIX)
    bin/
    etc/
    include/
    lib/
    share/
    .catkin
    env.bash
    setup.bash
    setup.sh
    ...
  install/                 -- INSTALL SPACE (set by CMAKE_INSTALL_PREFIX)
    bin/
    etc/
    include/
    lib/
    share/
    .catkin
    env.bash
    setup.bash
    setup.sh
    ...
```

catkin编译系统下的工作空间结构

1. ROS项目开发流程

创建工作空间

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

编译工作空间

```
$ cd ~/catkin_ws/
$ catkin_make
```

设置环境变量

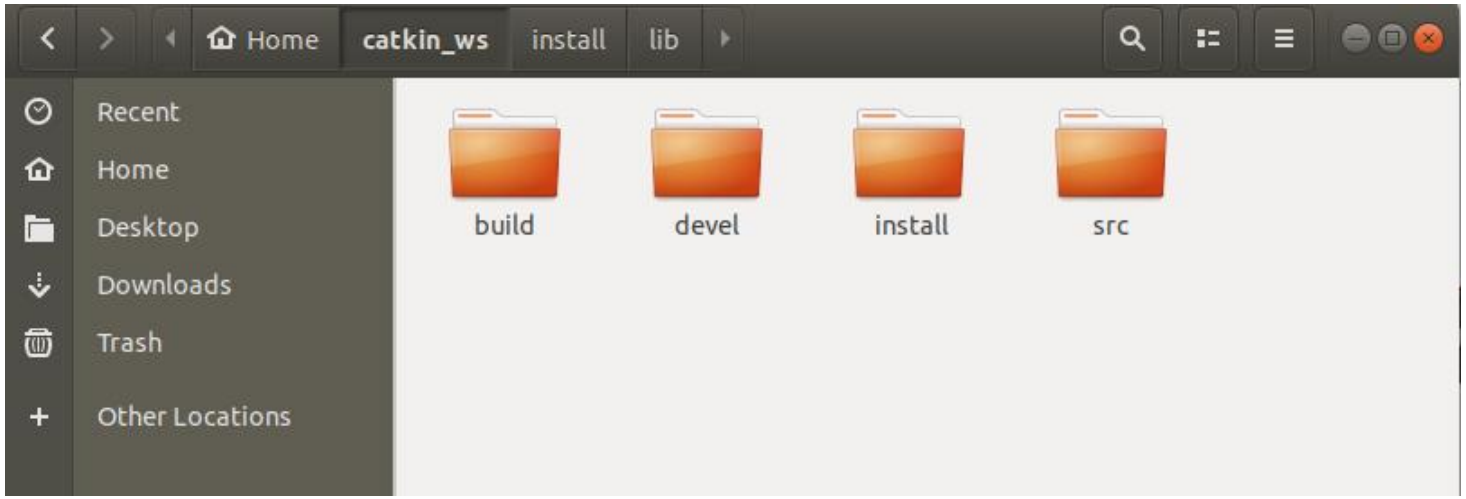
```
$ source devel/setup.bash
```

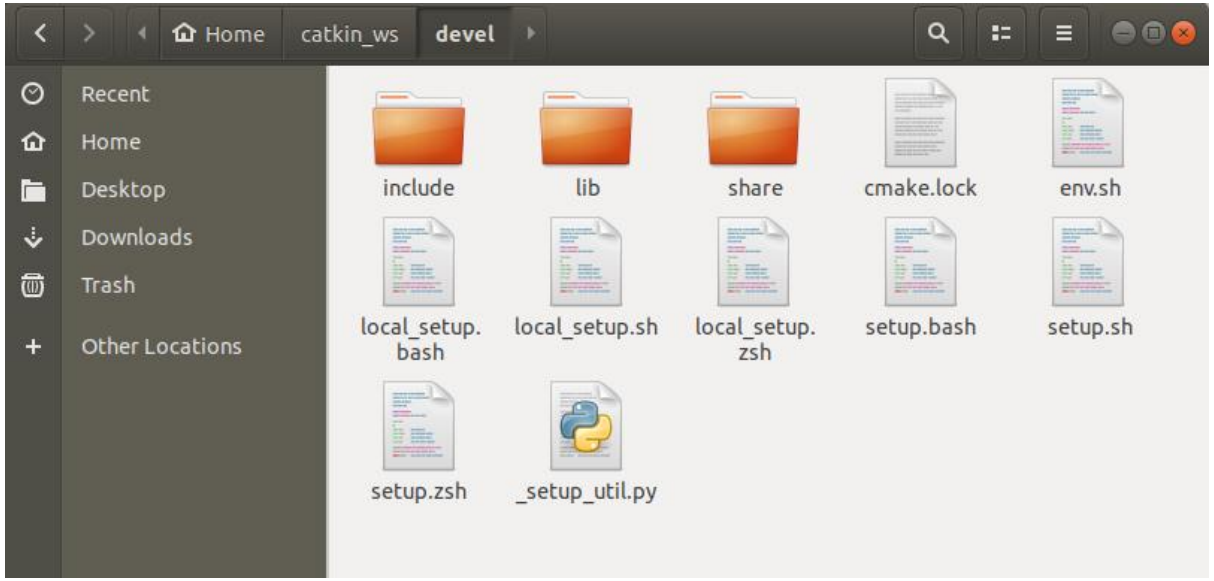
检查环境变量

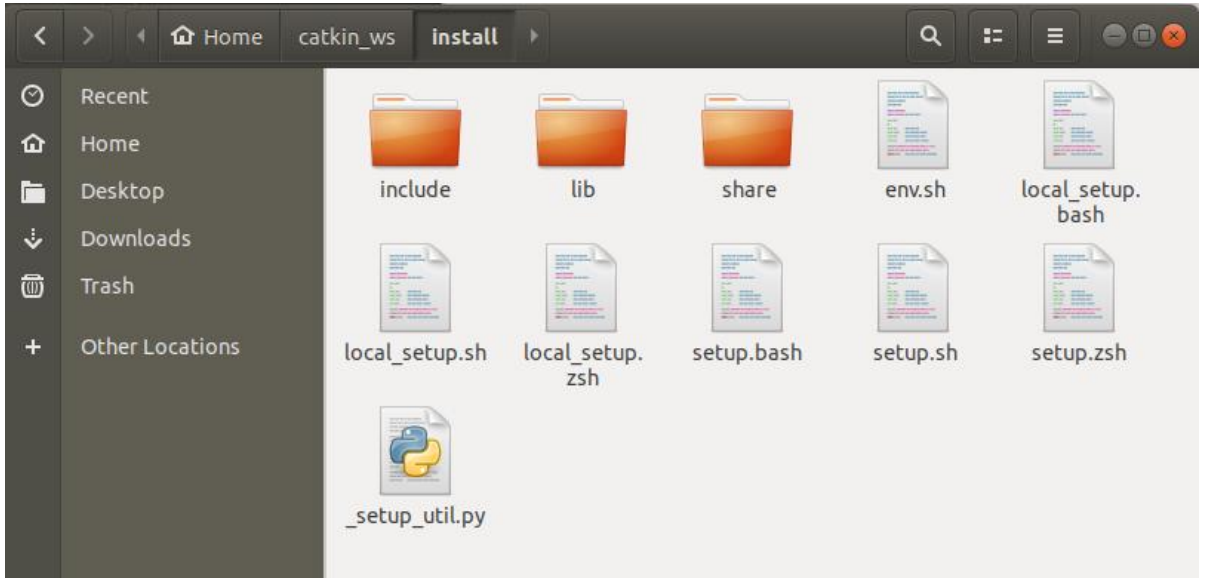
```
$ echo $ROS_PACKAGE_PATH
```

```
hcx@hcx-vpc:~$ echo $ROS_PACKAGE_PATH
/home/hcx/catkin_ws/src:/opt/ros/melodic/share
```

1. ROS项目开发流程


Home
catkin_ws
install
lib
Recent
Home
Desktop
Downloads
Trash
Other Locations
build
devel
install
src


Home
catkin_ws
devel
Recent
Home
Desktop
Downloads
Trash
Other Locations
include
lib
share
cmake.lock
env.sh
local_setup.bash
local_setup.sh
local_setup.zsh
setup.bash
setup.sh
setup.zsh
_setup_util.py


Home
catkin_ws
install
Recent
Home
Desktop
Downloads
Trash
Other Locations
include
lib
share
env.sh
local_setup.bash
local_setup.sh
local_setup.zsh
setup.bash
setup.sh
setup.zsh
_setup_util.py

1. ROS项目开发流程

$ catkin_create_pkg <package_name> [depend1] [depend2] [depend3]

创建功能包

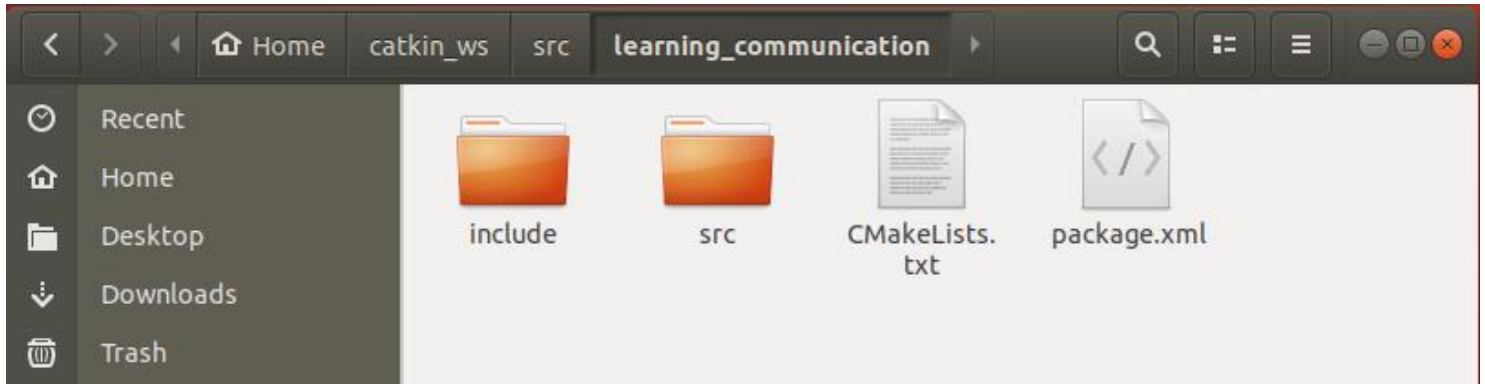
```
$ cd ~/catkin_ws/src
$ catkin_create_pkg learning_communication rospy roscpp std_msgs std_srvs
```

编译功能包

```
$ cd ~/catkin_ws
$ catkin_make
$ source ~/catkin_ws/devel/setup.bash
```

同一个工作空间下，不允许存在同名功能包
不同工作空间下，允许存在同名功能包

1. ROS项目开发流程



package.xml
~/catkin_ws/src/learning_communication

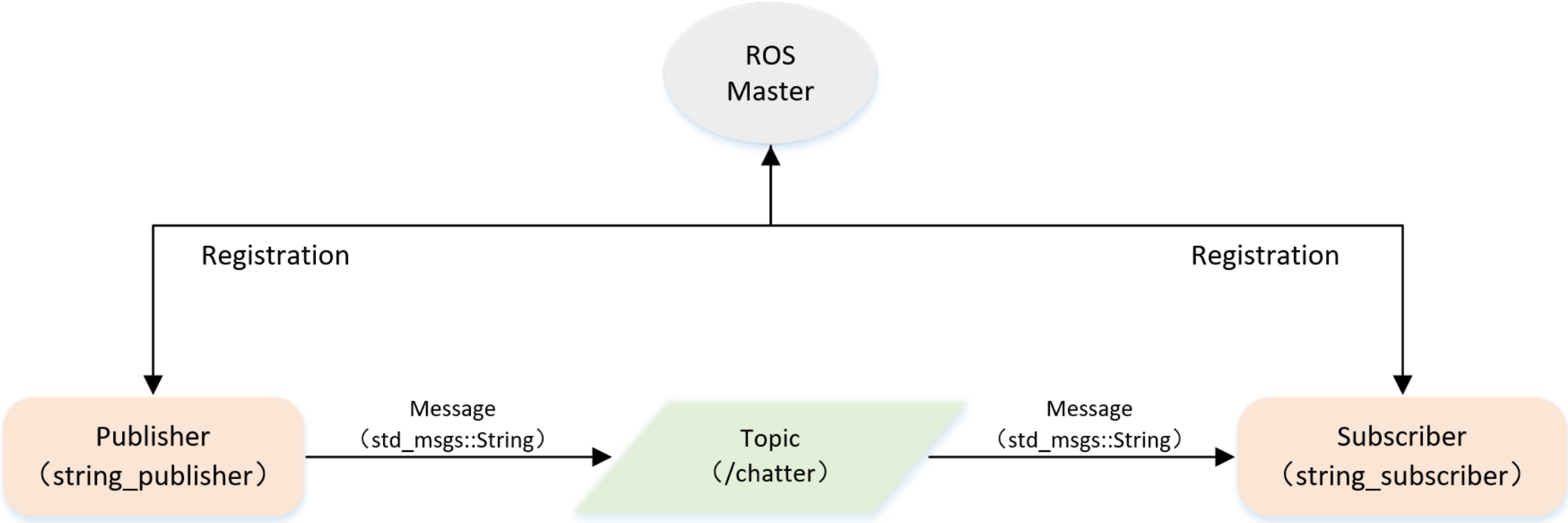
```xml
<?xml version="1.0"?>
<package format="2">
  <name>learning_communication</name>
  <version>0.0.0</version>
  <description>The learning_communication package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="hcx@todo.todo">hcx</maintainer>


  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>


  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/learning_communication</url> -->


  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->

```

CMakeLists.txt
~/catkin_ws/src/learning_communication

```
cmake_minimum_required(VERSION 2.8.3)
project(learning_communication)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  std_srvs
)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)


## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
# catkin_python_setup()

################################################
## Declare ROS messages, services and actions ##
################################################
```

2. ROS Topic通信编程

2. ROS Topic通信编程



话题模型（发布/订阅）

2. ROS Topic通信编程

```cpp
/**
 * 该例程将发布chatter话题，消息类型String
 */

#include <sstream>
#include "ros/ros.h"
#include "std_msgs/String.h"

int main(int argc, char **argv)
{
    // ROS节点初始化
    ros::init(argc, argv, "string_publisher");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个Publisher，发布名为chatter的topic，消息类型为std_msgs::String
    ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);

    // 设置循环的频率
    ros::Rate loop_rate(10);

    int count = 0;
    while (ros::ok())
    {
        // 初始化std_msgs::String类型的消息
        std_msgs::String msg;
        std::stringstream ss;
        ss << "hello world " << count;
        msg.data = ss.str();

        // 发布消息
        ROS_INFO("%s", msg.data.c_str());
        chatter_pub.publish(msg);

        // 按照循环频率延时
        loop_rate.sleep();
        ++count;
    }

    return 0;
}
```

string_publisher.cpp

如何实现一个发布者

- 初始化ROS节点；
- 向ROS Master注册节点信息，包括发布的话题名和话题中的消息类型；
- 创建消息数据；
- 按照一定频率循环发布消息。

2. ROS Topic通信编程

```cpp
/**
 * 该例程将订阅chatter话题，消息类型String
 */

#include "ros/ros.h"
#include "std_msgs/String.h"

// 接收到订阅的消息后，会进入消息回调函数
void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
    // 将接收到的消息打印出来
    ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv)
{
    // 初始化ROS节点
    ros::init(argc, argv, "string_subscriber");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个Subscriber，订阅名为chatter的topic，注册回调函数chatterCallback
    ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

    // 循环等待回调函数
    ros::spin();

    return 0;
}
```

string_subscriber.cpp

如何实现一个订阅者

- 初始化ROS节点；
- 订阅需要的话题；
- 循环等待话题消息，接收到消息后进入回调函数；
- 在回调函数中完成消息处理。

2. ROS Topic通信编程

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_communication_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(string_publisher src/string_publisher.cpp)
target_link_libraries(string_publisher ${catkin_LIBRARIES})

add_executable(string_subscriber src/string_subscriber.cpp)
target_link_libraries(string_subscriber ${catkin_LIBRARIES})
```

如何配置CMakeLists.txt中的编译规则

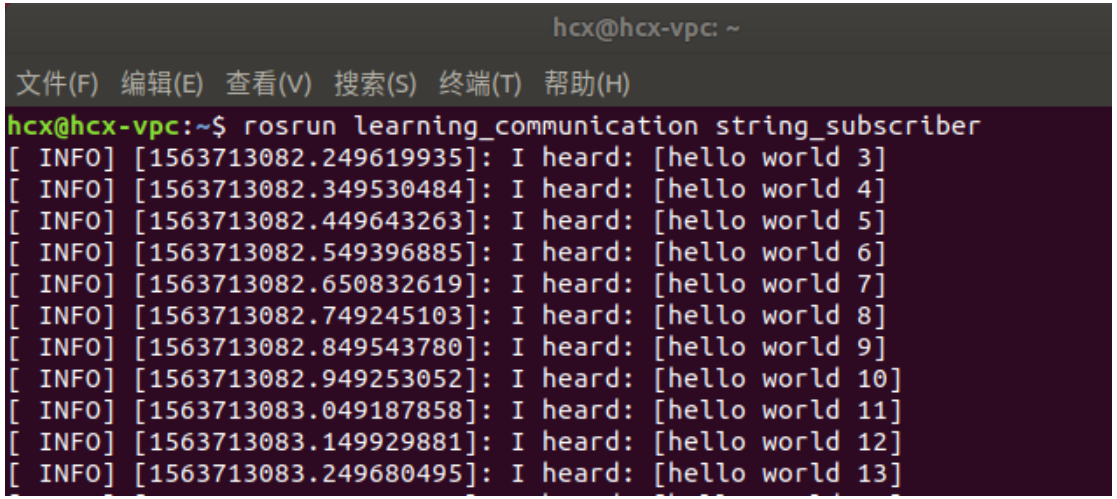
- 设置需要编译的代码和生成的可执行文件；
- 设置链接库；

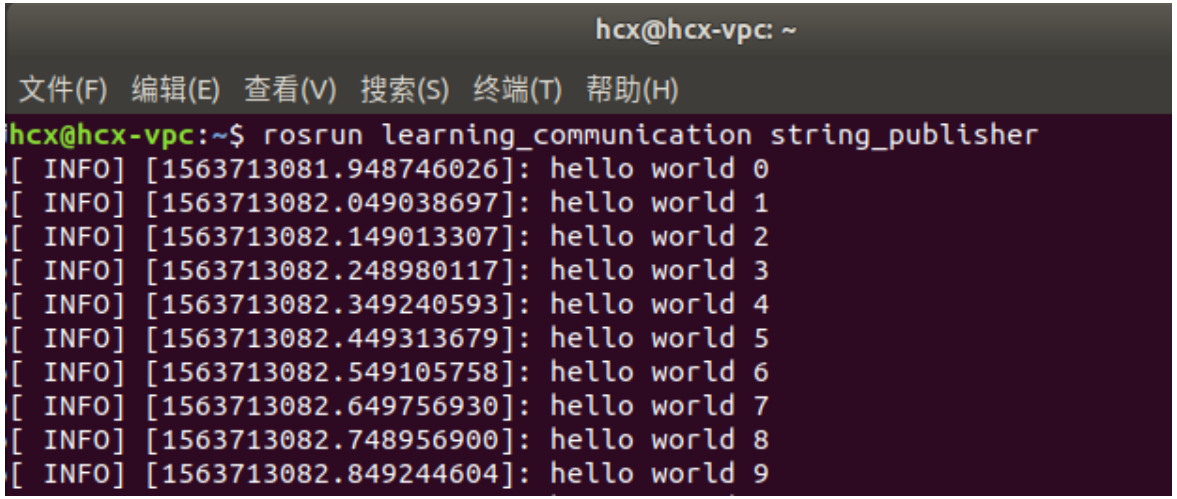
```
add_executable(string_publisher src/string_publisher.cpp)
target_link_libraries(string_publisher ${catkin_LIBRARIES})

add_executable(string_subscriber src/string_subscriber.cpp)
target_link_libraries(string_subscriber ${catkin_LIBRARIES})
```

CMakeLists.txt

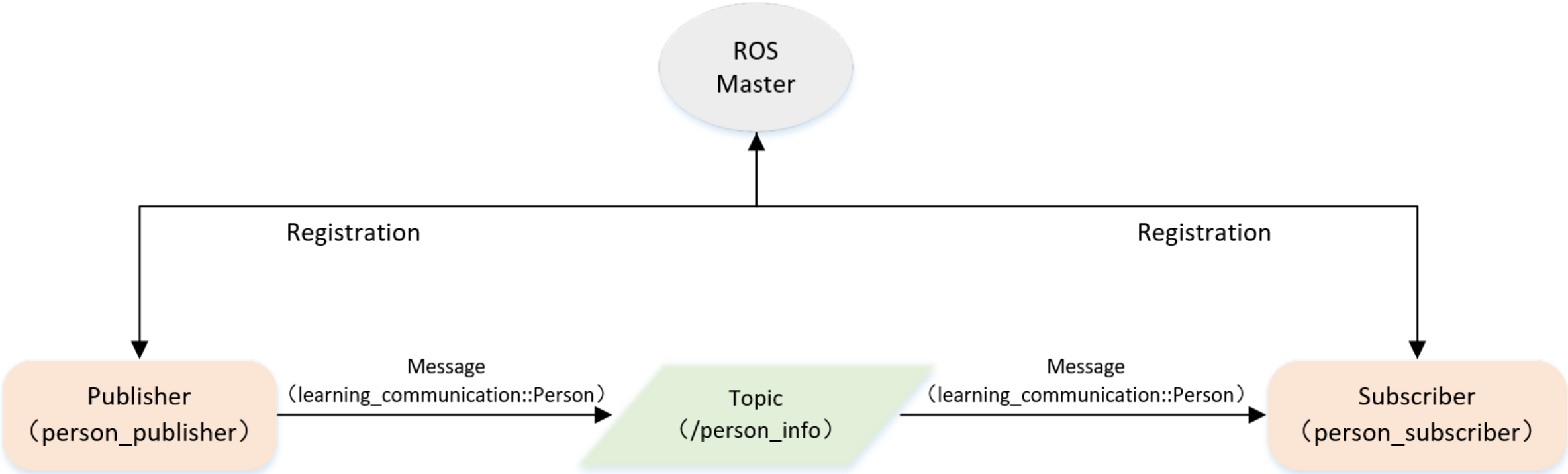
2. ROS Topic通信编程

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun learning_communication string_publisher
$ rosrun learning_communication string_subscriber
```





2. ROS Topic通信编程



话题模型（发布/订阅）

2. ROS Topic通信编程

如何自定义话题消息

```
string name
uint8  sex
uint8  age

uint8 unknown = 0
uint8 male    = 1
uint8 female  = 2
```

PersonMsg.msg

- **定义msg文件；**
- **在package.xml中添加功能包依赖**

  <build_depend>message_generation</build_depend>
  <exec_depend>message_runtime</exec_depend>

- **在CMakeLists.txt添加编译选项**
  - find_package(…… message_generation)
  - add_message_files(FILES PersonMsg.msg)
    generate_messages(DEPENDENCIES std_msgs)
  - catkin_package(…… message_runtime)
- **编译生成语言相关文件**

2. ROS Topic通信编程

```cpp
/**
 * 该例程将发布/person_info话题，learning_communication::PersonMsg
 */

#include <ros/ros.h>
#include "learning_communication/PersonMsg.h"

int main(int argc, char **argv)
{
    // ROS节点初始化
    ros::init(argc, argv, "person_publisher");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个Publisher，发布名为/person_info的topic，消息类型为learning_communication::PersonMsg，队列长度10
    ros::Publisher person_info_pub = n.advertise<learning_communication::PersonMsg>("/person_info", 10);

    // 设置循环的频率
    ros::Rate loop_rate(1);

    int count = 0;
    while (ros::ok())
    {
        // 初始化learning_communication::Person类型的消息
        learning_communication::PersonMsg person_msg;
        person_msg.name = "Tom";
        person_msg.age  = 18;
        person_msg.sex  = learning_communication::PersonMsg::male;

        // 发布消息
        person_info_pub.publish(person_msg);

        ROS_INFO("Publish Person Info: name:%s  age:%d  sex:%d",
                 person_msg.name.c_str(), person_msg.age, person_msg.sex);

        // 按照循环频率延时
        loop_rate.sleep();
    }

    return 0;
}
```

person_publisher.cpp

如何实现一个发布者

- 初始化ROS节点；
- 向ROS Master注册节点信息，包括发布的话题名和话题中的消息类型；
- 创建消息数据；
- 按照一定频率循环发布消息。

2. ROS Topic通信编程

```cpp
/**
 * 该例程将订阅/person_info话题，自定义消息类型learning_communication::PersonMsg
 */

#include <ros/ros.h>
#include "learning_communication/PersonMsg.h"

// 接收到订阅的消息后，会进入消息回调函数
void personInfoCallback(const learning_communication::PersonMsg::ConstPtr& msg)
{
    // 将接收到的消息打印出来
    ROS_INFO("Subcribe Person Info: name:%s  age:%d  sex:%d",
             msg->name.c_str(), msg->age, msg->sex);
}

int main(int argc, char **argv)
{
    // 初始化ROS节点
    ros::init(argc, argv, "person_subscriber");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个Subscriber，订阅名为/person_info的topic，注册回调函数personInfoCallback
    ros::Subscriber person_info_sub = n.subscribe("/person_info", 10, personInfoCallback);

    // 循环等待回调函数
    ros::spin();

    return 0;
}
```

person_subscriber.cpp

如何实现一个订阅者

- 初始化ROS节点；
- 订阅需要的话题；
- 循环等待话题消息，接收到消息后进入回调函数；
- 在回调函数中完成消息处理。

2. ROS Topic通信编程

古月居 GYH.AI

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_communication_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

## Add cmake target dependencies of the executable
## same as for the library above
# add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

add_executable(person_publisher src/person_publisher.cpp)
target_link_libraries(person_publisher ${catkin_LIBRARIES})
add_dependencies(person_publisher ${PROJECT_NAME}_gencpp)

add_executable(person_subscriber src/person_subscriber.cpp)
target_link_libraries(person_subscriber ${catkin_LIBRARIES})
add_dependencies(person_subscriber ${PROJECT_NAME}_gencpp)
```

如何配置CMakeLists.txt中的编译规则

- 设置需要编译的代码和生成的可执行文件；
- 设置链接库；
- 添加依赖项。

```
add_executable(person_publisher src/person_publisher.cpp)
target_link_libraries(person_publisher ${catkin_LIBRARIES})
add_dependencies(person_publisher ${PROJECT_NAME}_gencpp)

add_executable(person_subscriber src/person_subscriber.cpp)
target_link_libraries(person_subscriber ${catkin_LIBRARIES})
add_dependencies(person_subscriber ${PROJECT_NAME}_gencpp)
```

CMakeLists.txt

2. ROS Topic通信编程

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun learning_communication person_subscriber
$ rosrun learning_communication person_publisher
```

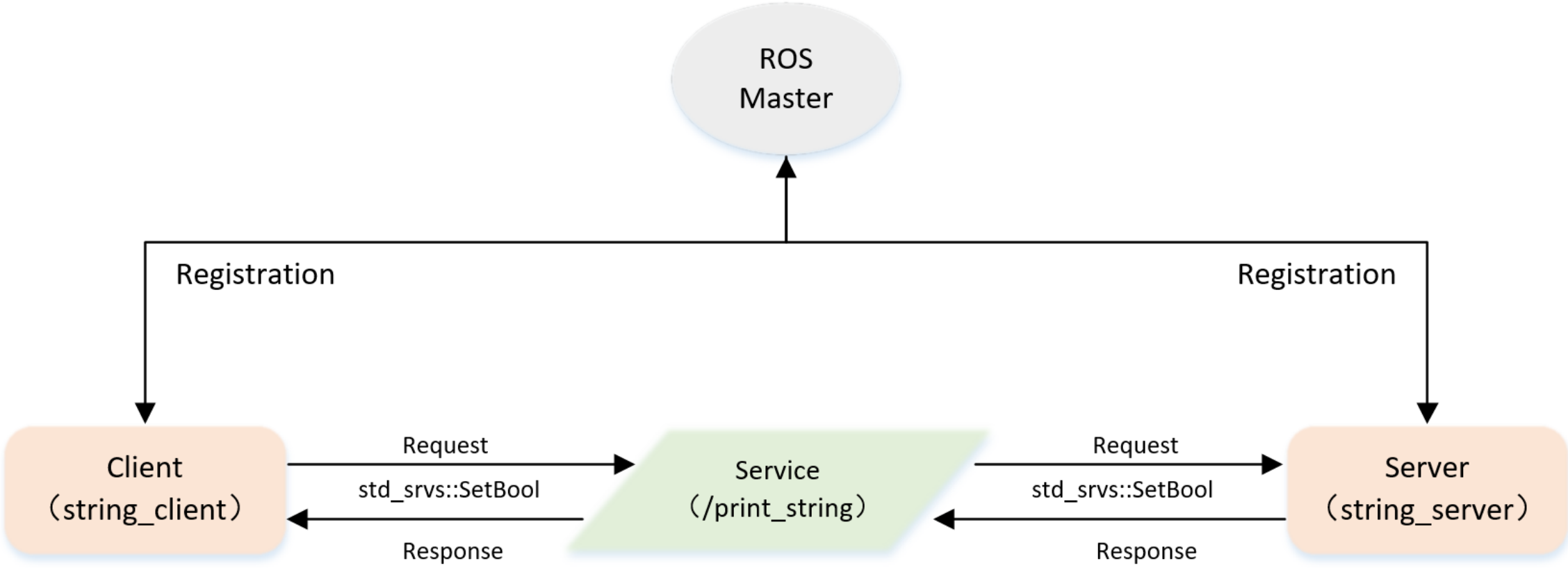
```
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_topic person_subscriber
[ INFO] [1562216316.857673702]: Subcribe Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216317.857324485]: Subcribe Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216318.857310636]: Subcribe Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216319.856921435]: Subcribe Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216320.856461694]: Subcribe Person Info: name:Tom  age:18  sex:1
```

```
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_topic person_publisher
[ INFO] [1562216315.855698333]: Publish Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216316.856484874]: Publish Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216317.856251972]: Publish Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216318.856513919]: Publish Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216319.856089664]: Publish Person Info: name:Tom  age:18  sex:1
[ INFO] [1562216320.855924037]: Publish Person Info: name:Tom  age:18  sex:1
```

古月居 GYH.AI

3. ROS Service通信编程

3. ROS Service通信编程



服务模型（服务端/客户端）

3. ROS Service通信编程

```cpp
/**
 * 该例程将提供print_string服务，std_srvs::SetBool
 */

#include "ros/ros.h"
#include "std_srvs/SetBool.h"

// service回调函数，输入参数req，输出参数res
bool print(std_srvs::SetBool::Request  &req,
           std_srvs::SetBool::Response &res)
{
    // 打印字符串
    if(req.data)
    {
        ROS_INFO("Hello ROS!");
        res.success = true;
        res.message = "Print Successully";
    }
    else
    {
        res.success = false;
        res.message = "Print Failed";
    }

    return true;
}

int main(int argc, char **argv)
{
    // ROS节点初始化
    ros::init(argc, argv, "string_server");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个名为print_string的server，注册回调函数print()
    ros::ServiceServer service = n.advertiseService("print_string", print);

    // 循环等待回调函数
    ROS_INFO("Ready to print hello string.");
    ros::spin();

    return 0;
}
```

string_server.cpp

如何实现一个服务器

- 初始化ROS节点；
- 创建Server实例；
- 循环等待服务请求，进入回调函数；
- 在回调函数中完成服务功能的处理，并反馈应答数据。

3. ROS Service通信编程

```cpp
/**
 * 该例程将请求print_string服务，std_srvs::SetBool
 */

#include "ros/ros.h"
#include "std_srvs/SetBool.h"

int main(int argc, char **argv)
{
    // ROS节点初始化
    ros::init(argc, argv, "string_client");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个client，service消息类型是std_srvs::SetBool
    ros::ServiceClient client = n.serviceClient<std_srvs::SetBool>("print_string");

    // 创建std_srvs::SetBool类型的service消息
    std_srvs::SetBool srv;
    srv.request.data = true;

    // 发布service请求，等待应答结果
    if (client.call(srv))
    {
        ROS_INFO("Response : [%s] %s", srv.response.success?"True":"False",
                                       srv.response.message.c_str());
    }
    else
    {
        ROS_ERROR("Failed to call service print_string");
        return 1;
    }

    return 0;
}
```

string_client.cpp

如何实现一个客户端

- 初始化ROS节点；
- 创建一个Client实例；
- 发布服务请求数据；
- 等待Server处理之后的应答结果。

3. ROS Service通信编程

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_communication_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(string_server src/string_server.cpp)
target_link_libraries(string_server ${catkin_LIBRARIES})

add_executable(string_client src/string_client.cpp)
target_link_libraries(string_client ${catkin_LIBRARIES})
```

如何配置CMakeLists.txt中的编译规则

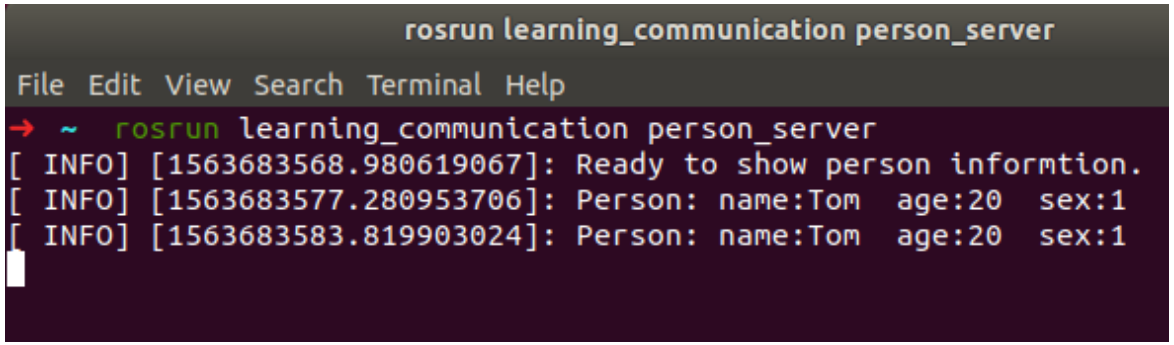
- 设置需要编译的代码和生成的可执行文件；
- 设置链接库；

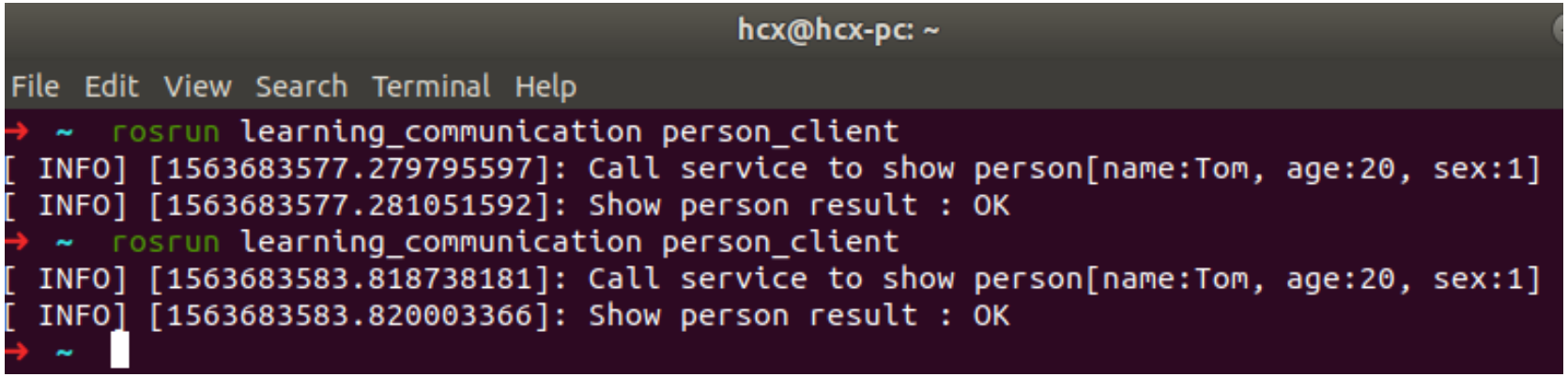
```
add_executable(string_server src/string_server.cpp)
target_link_libraries(string_server ${catkin_LIBRARIES})

add_executable(string_client src/string_client.cpp)
target_link_libraries(string_client ${catkin_LIBRARIES})
```

CMakeLists.txt

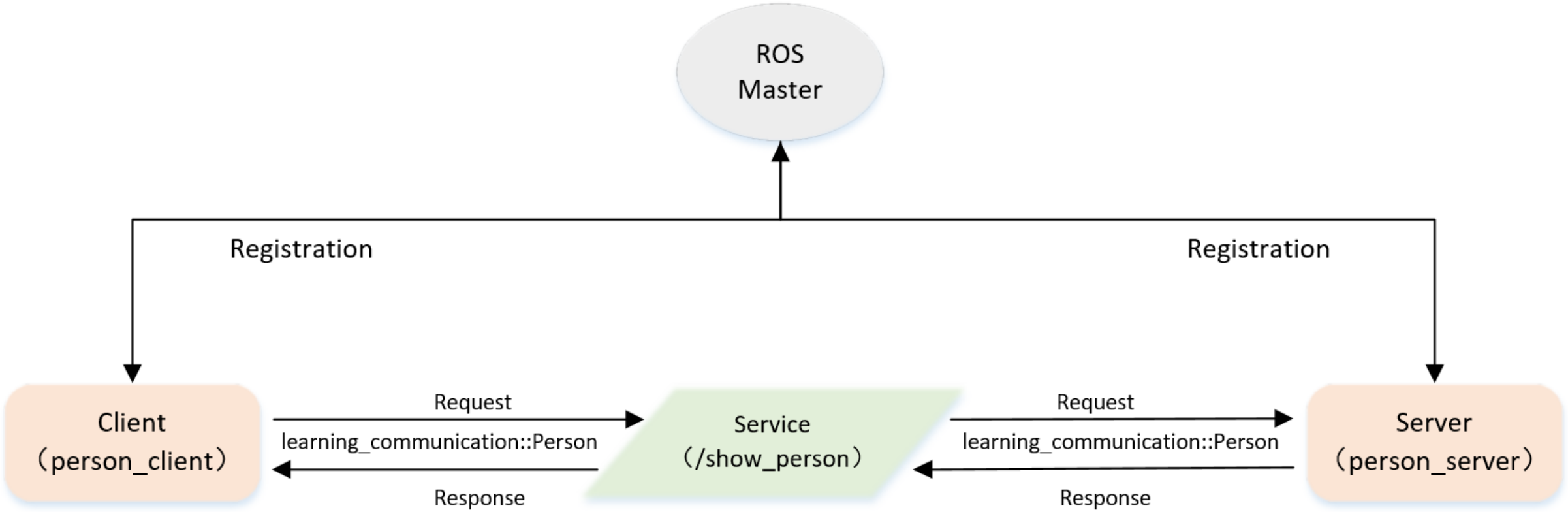
3. ROS Service通信编程

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun learning_communication string_server
$ rosrun learning_communication string_client
```





3. ROS Service通信编程



服务模型（服务端/客户端）

3. ROS Service通信编程

如何自定义服务数据

```
string name
uint8  age
uint8  sex

uint8 unknown = 0
uint8 male    = 1
uint8 female  = 2
---
string result
```

PersonSrv.srv

- **定义srv文件；**
- **在package.xml中添加功能包依赖**

  <build_depend>message_generation</build_depend>
  <exec_depend>message_runtime</exec_depend>

- **在CMakeLists.txt添加编译选项**
  - find_package(…… message_generation)
  - add_service_files(FILES PersonSrv.srv)
    generate_messages(DEPENDENCIES std_msgs)
  - catkin_package(…… message_runtime)
- **编译生成语言相关文件**

3. ROS Service通信编程

```cpp
/**
 * 该例程将执行/show_person服务，服务数据类型learning_communication::PersonSrv
 */

#include <ros/ros.h>
#include "learning_communication/PersonSrv.h"

// service回调函数，输入参数req，输出参数res
bool personCallback(learning_communication::PersonSrv::Request  &req,
                    learning_communication::PersonSrv::Response &res)
{
    // 显示请求数据
    ROS_INFO("Person: name:%s  age:%d  sex:%d", req.name.c_str(), req.age, req.sex);

    // 设置反馈数据
    res.result = "OK";

    return true;
}

int main(int argc, char **argv)
{
    // ROS节点初始化
    ros::init(argc, argv, "person_server");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个名为/show_person的server，注册回调函数personCallback
    ros::ServiceServer person_service = n.advertiseService("/show_person", personCallback);

    // 循环等待回调函数
    ROS_INFO("Ready to show person informtion.");
    ros::spin();

    return 0;
}
```

person_server.cpp

如何实现一个服务器

- 初始化ROS节点；
- 创建Server实例；
- 循环等待服务请求，进入回调函数；
- 在回调函数中完成服务功能的处理，并反馈应答数据。

3. ROS Service通信编程

古月居 GYH.AI

```
/**
 * 该例程将请求/show_person服务，服务数据类型learning_communication::PersonSrv
 */

#include <ros/ros.h>
#include "learning_communication/PersonSrv.h"

int main(int argc, char** argv)
{
    // 初始化ROS节点
    ros::init(argc, argv, "person_client");

    // 创建节点句柄
    ros::NodeHandle node;

    // 发现/spawn服务后，创建一个服务客户端，连接名为/spawn的service
    ros::service::waitForService("/show_person");
    ros::ServiceClient person_client = node.serviceClient<learning_communication::PersonSrv>("/show_person");

    // 初始化learning_communication::Person的请求数据
    learning_communication::PersonSrv srv;
    srv.request.name = "Tom";
    srv.request.age  = 20;
    srv.request.sex  = learning_communication::PersonSrv::Request::male;

    // 请求服务调用
    ROS_INFO("Call service to show person[name:%s, age:%d, sex:%d]",
             srv.request.name.c_str(), srv.request.age, srv.request.sex);

    person_client.call(srv);

    // 显示服务调用结果
    ROS_INFO("Show person result : %s", srv.response.result.c_str());

    return 0;
};
```

person_client.cpp

如何实现一个客户端

- 初始化ROS节点；
- 创建一个Client实例；
- 发布服务请求数据；
- 等待Server处理之后的应答结果。

3. ROS Service通信编程

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_communication_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

## Add cmake target dependencies of the executable
## same as for the library above
# add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})


add_executable(person_server src/person_server.cpp)
target_link_libraries(person_server ${catkin_LIBRARIES})
add_dependencies(person_server ${PROJECT_NAME}_gencpp)

add_executable(person_client src/person_client.cpp)
target_link_libraries(person_client ${catkin_LIBRARIES})
add_dependencies(person_client ${PROJECT_NAME}_gencpp)
```

如何配置CMakeLists.txt中的编译规则

- 设置需要编译的代码和生成的可执行文件；
- 设置链接库；
- 添加依赖项。

```
add_executable(person_server src/person_server.cpp)
target_link_libraries(person_server ${catkin_LIBRARIES})
add_dependencies(person_server ${PROJECT_NAME}_gencpp)

add_executable(person_client src/person_client.cpp)
target_link_libraries(person_client ${catkin_LIBRARIES})
add_dependencies(person_client ${PROJECT_NAME}_gencpp)
```

CMakeLists.txt

3. ROS Service通信编程

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun learning_communication person_server
$ rosrun learning_communication person_client
```

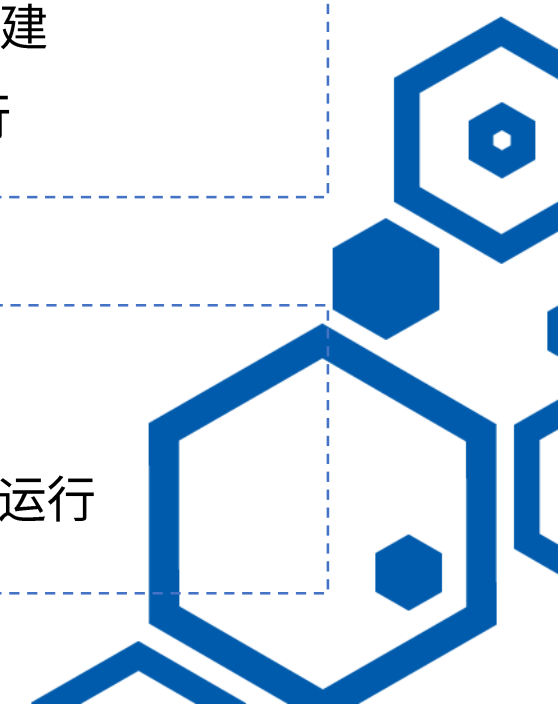
```
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_service person_server
[ INFO] [1562234385.473929292]: Ready to show person informtion.
[ INFO] [1562234405.584154235]: Person: name:Tom  age:20  sex:1
[ INFO] [1562234411.809871741]: Person: name:Tom  age:20  sex:1
```

```
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_service person_client
[ INFO] [1562234405.582071660]: Call service to show person[name:Tom, age:20, se
x:1]
[ INFO] [1562234405.584514656]: Show person result : OK
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_service person_client
[ INFO] [1562234411.808122249]: Call service to show person[name:Tom, age:20, se
x:1]
[ INFO] [1562234411.810180819]: Show person result : OK
```

本讲小结

古月居 GYH.AI

ROS项目开发流程	工作空间 → 功能包 → 源代码 → 编译 → 运行
ROS Topic通信编程	发布者Publisher、订阅者Subscriber的创建 自定义话题消息、编译规则的设置与运行
ROS Service通信编程	服务器Server、客户端Client的创建 自定义请求/应答数据、编译规则的设置与运行



本讲作业

使用海龟仿真器，完成以下编程作业：

1. 创建一个节点，在其中实现一个订阅者和一个发布者，完成以下功能：

- 发布者：发布海龟速度指令，让海龟圆周运动
- 订阅者：订阅海龟的位置信息，并在终端中周期打印输出

2. 创建另外一个节点，在其中实现一个客户端，完成以下功能：

- 客户端：请求海龟诞生的服务，在仿真器中产生一只新的海龟

3. 综合运用话题与服务编程、命令行使用，实现以下场景：

小R想要实现一个海龟运动控制的功能包， 需要具备以下功能（以下指令的接收方均为该功能包中的节点）：

- 通过命令行发送新生海龟的名字，即可在界面中产生一只海龟，并且位置不重叠；
- 通过命令行发送指令控制界面中任意海龟圆周运动的启动/停止，速度可通过命令行控制；

你可以帮助小R实现这个功能包么？

扩展阅读

古月居 GYH.AI

- 古月·ROS入门21讲
https://www.bilibili.com/video/av59458869

- A Gentle Introduction to ROS
http://wiki.ros.org/Books/AGentleIntroductiontoROS_Chinese

- ROS Tutorials
http://wiki.ros.org/ROS/Tutorials

- ROS APIs (C++ / Python)
http://wiki.ros.org/APIs

- 《ROS机器人开发实践》
第3章

Thank You

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居

古月春旭