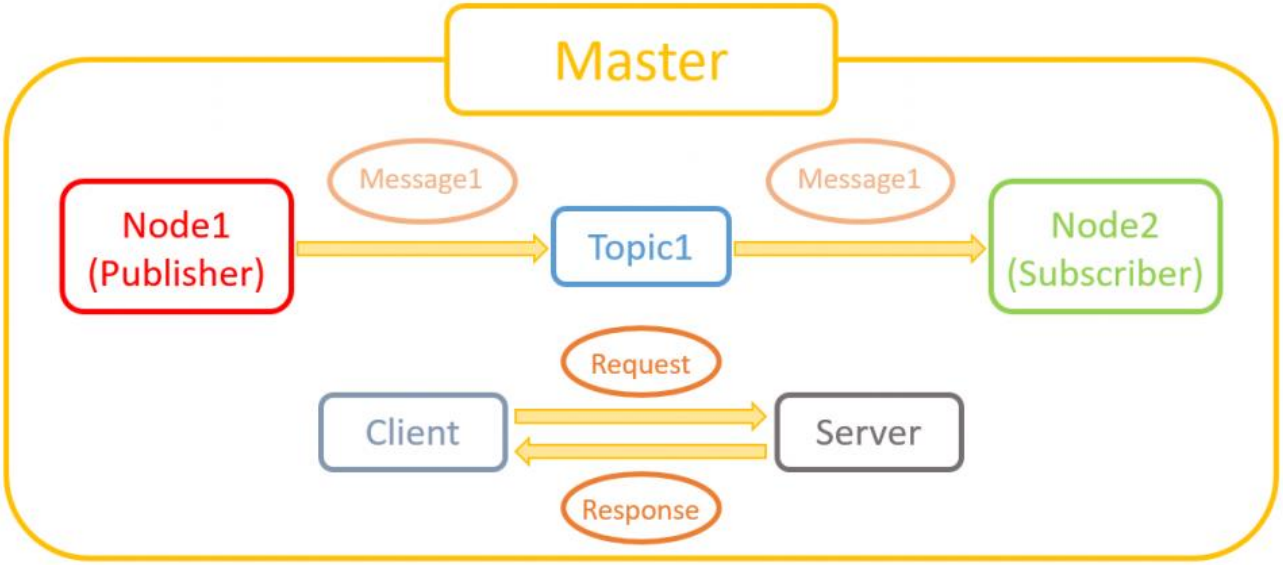
無人載具技術與應用
ROS

徐瑋隆
wlhsu304@gmail.com

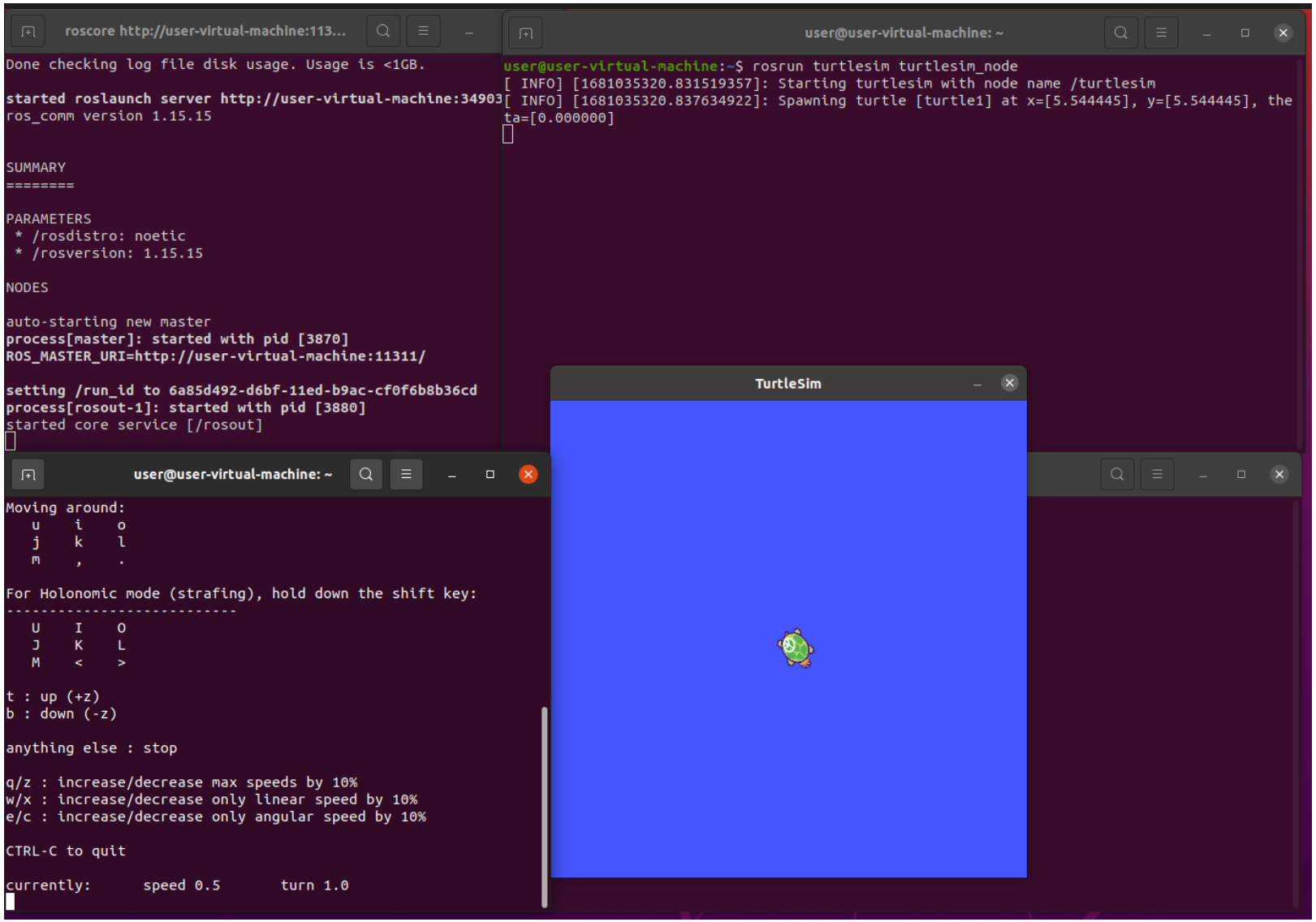
ROS SERVICE

ROS Service



- Services 同時包含了 Request 及 Response 的訊息機制
  - 在 Topic 裡頭有 Publisher 以及 Subscriber，兩者是由 Publisher 向 Topic 發布訊息，Subscriber 才會接收到訊息。
  - Service 裡只有 Server 與 Client，Client 必須透過 Service 向 Server 提出 Request，而 Server 才會回傳對應的 Response 給 Client。

rosservice – turtlesim 01



roscore

rosrun turtlesim turtlesim_node

rosrun teleop_twist_keyboard teleop_twist_keyboard.py /cmd_vel:=/turtle1/cmd_vel

rosservice – turtlesim 02

rostopic list

rosservice list

rosservice call /reset

rosservice type /reset | rossrv show

rosservice type /spawn | rossrv show

rosservice info /spawn

rosservice call /spawn 1 2 0 kk

```
user@user-virtual-machine:~$ rosservice list
/clear
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/spawn
/teleop_twist_keyboard/get_loggers
/teleop_twist_keyboard/set_logger_level
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/get_loggers
/turtlesim/set_logger_level
user@user-virtual-machine:~$ rosservice call /reset

user@user-virtual-machine:~$ rosservice call /reset

user@user-virtual-machine:~$ rosservice type /spawn | rossrv show
float32 x
float32 y
float32 theta
string name
---
string name

user@user-virtual-machine:~$ rosservice call /spawn "x: 1.0
y: 3.0
theta: 1.0
name: 'abc'"
name: "abc"
```

rosservice – turtlesim 03

rostopic list

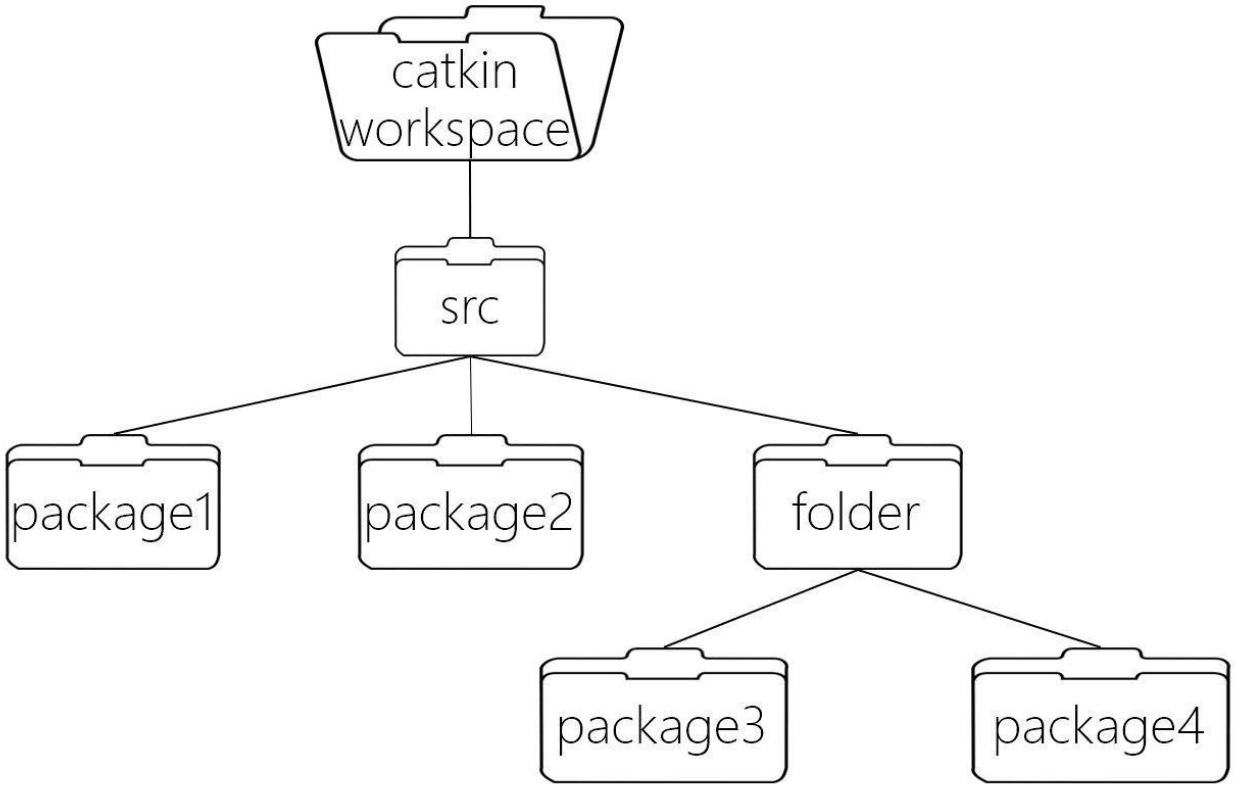
rosservice info /kill

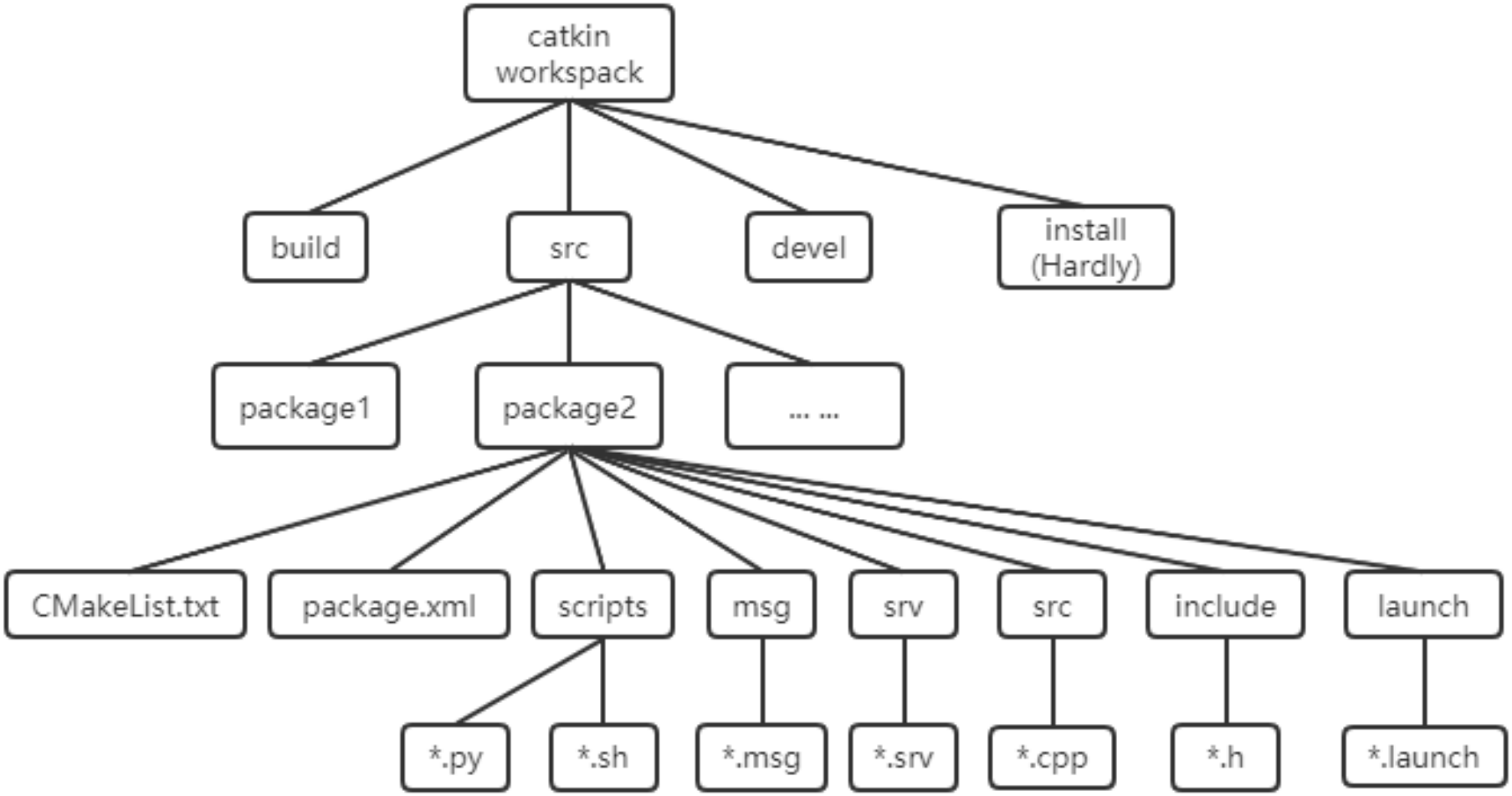
rosservice call /kill kk

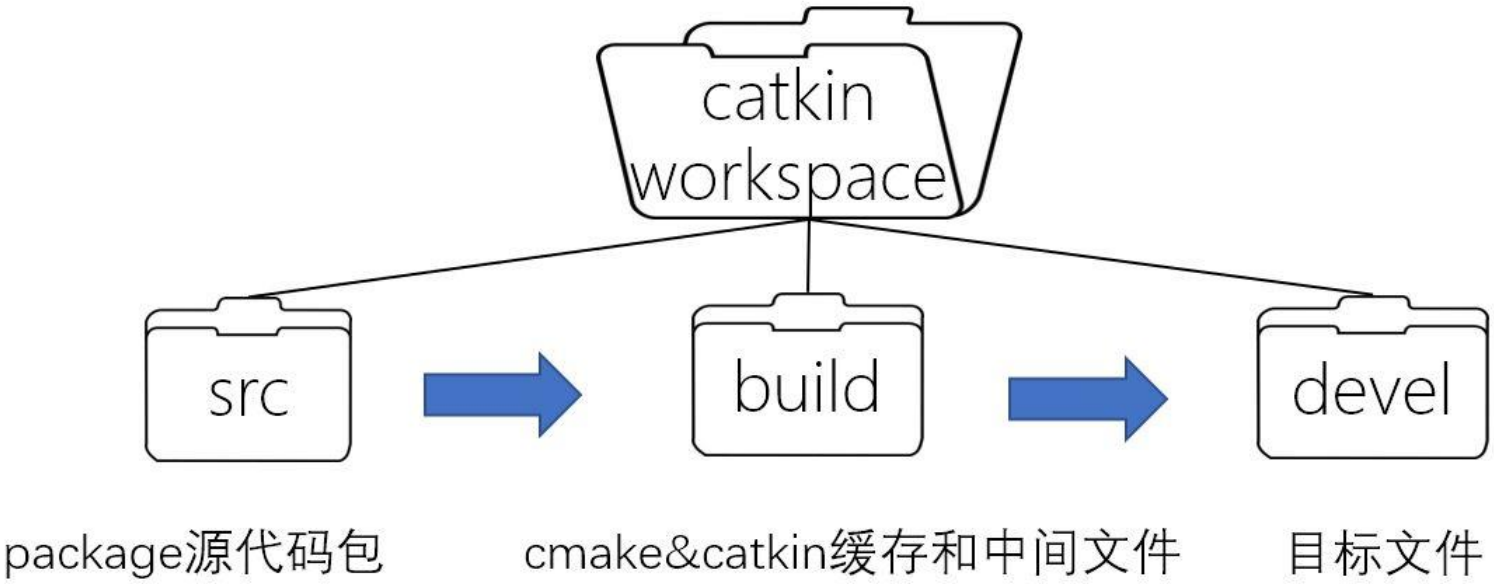
```
user@user-virtual-machine:~$ rostopic list
/kk/cmd_vel
/kk/color_sensor
/kk/pose
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
user@user-virtual-machine:~$ rosservice call /kill kk

user@user-virtual-machine:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

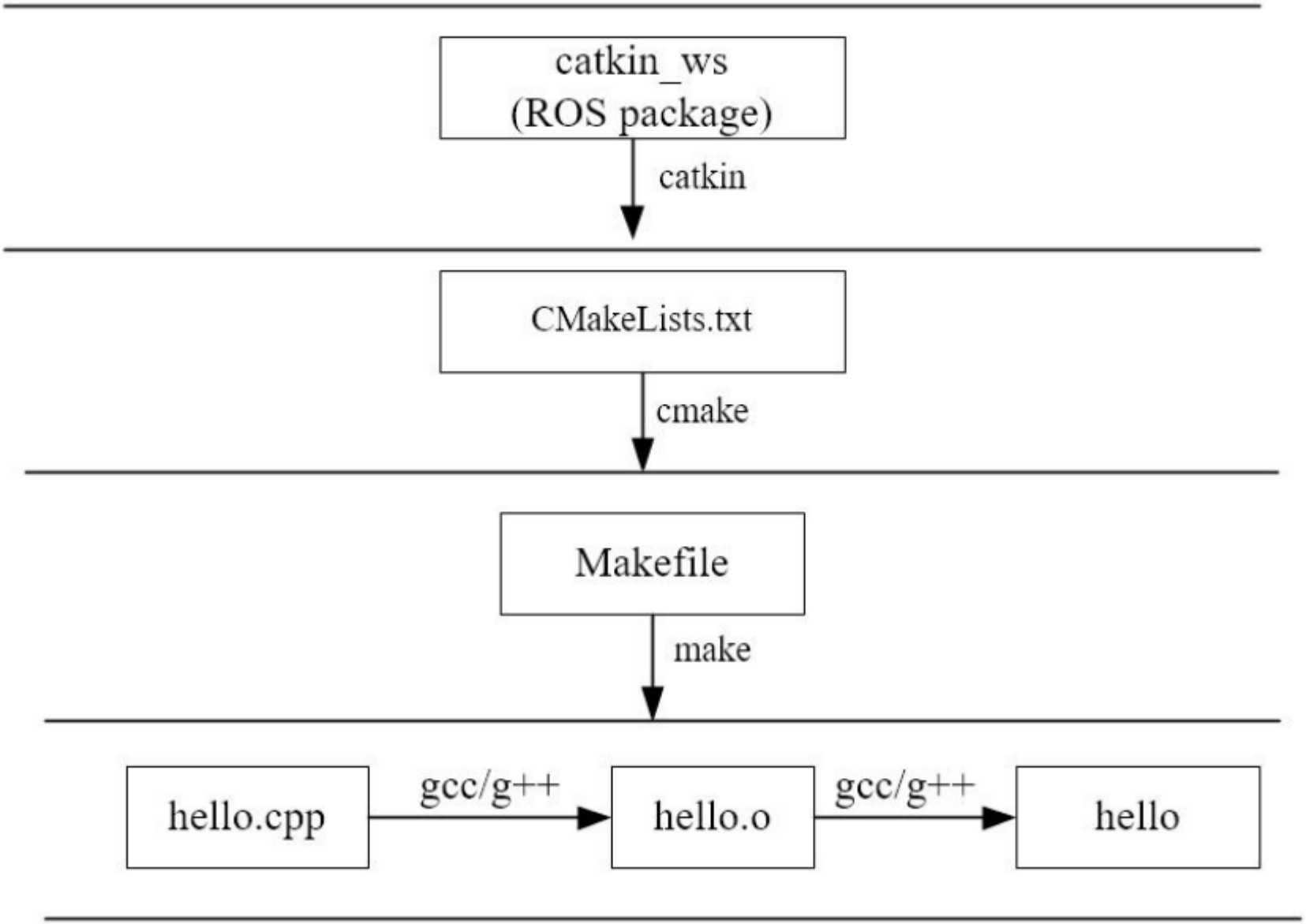
ros node coding

ROS SERVICE


catkin
workspace
src
package1
package2
folder
package3
package4


catkin workspack
build
src
devel
install (Hardly)
package1
package2
... ...
CMakeList.txt
package.xml
scripts
msg
srv
src
include
launch
*.py
*.sh
*.msg
*.srv
*.cpp
*.h
*.launch


catkin
workspace
src
build
devel
package源代码包
cmake&catkin缓存和中间文件
目标文件


catkin_ws
(ROS package)
catkin
CMakeLists.txt
cmake
Makefile
make
hello.cpp
gcc/g++
hello.o
gcc/g++
hello

ros node 01

```
user@user-virtual-machine:~$ cd catkin_ws/src/
user@user-virtual-machine:~/catkin_ws/src$ catkin_create_pkg my_work03 roscpp ro
spy std_msgs
Created file my_work03/package.xml
Created file my_work03/CMakeLists.txt
Created folder my_work03/include/my_work03
Created folder my_work03/src
Successfully created files in /home/user/catkin_ws/src/my_work03. Please adjust
the values in package.xml.
```

cd catkin_ws/src/
catkin_create_pkg my_work03 roscpp rospy std_msgs
ls

ros node 02

```
user@user-virtual-machine:~/catkin_ws/src$ cd my_work03/src/
user@user-virtual-machine:~/catkin_ws/src/my_work03/src$ gedit myNode01.cpp &
```

Open | myNode01.cpp ~/catkin_ws/src/my_work03/src | Save

```cpp
#include "ros/ros.h"

int main(int argc, char **argv)
{
  ros::init(argc, argv, "myNode01");
  ros::NodeHandle n;

  ros::Rate loop_rate(10);   // 10Hz

  while (ros::ok())
  {
    ros::spinOnce();
    loop_rate.sleep();
  }
  return 0;
}
```

cd my_work03/src/
gedit myNode01.cpp &

ros node 03

```
user@user-virtual-machine:~/catkin_ws/src$ cd my_work02/src/
user@user-virtual-machine:~/catkin_ws/src/my_work02/src$ gedit myNode01.cpp
```

Open | myNode01.cpp ~/catkin_ws/src/my_work02/src | Save

```
#include "ros/ros.h"

int main(int argc, char **argv)
{
  ros::init(argc, argv, "myNode01");
  ros::NodeHandle n;

  ros::Rate loop_rate(10);   // 10Hz

  while (ros::ok())
  {
    ros::spinOnce();
    loop_rate.sleep();
  }
  return 0;
}
```

```
#include "ros/ros.h"
int main(int argc, char **argv)
{
  ros::init(argc, argv, "myNode01");
  ros::NodeHandle n;
  ros::Rate loop_rate(10);   // 10Hz
  ROS_INFO("myNode01: hi");
  while (ros::ok())  {
    ros::spinOnce();
    loop_rate.sleep();
  }
  return 0;
}
```

ros node 04

```
user@user-virtual-machine:~/catkin_ws/src/my_work03/src$ cd ..
user@user-virtual-machine:~/catkin_ws/src/my_work03$ gedit CMakeLists.txt &
[2] 24554
user@user-virtual-machine:~/catkin_ws/src/my_work03$
```

Open | CMakeLists.txt ~/catkin_ws/src/my_work03 | Save

myNode01.cpp | CMakeLists.txt

```
## Build ##
###########

## Specify additional locations of header files
## Your package locations should be listed before other locations
include_directories(
# include
  ${catkin_INCLUDE_DIRS}
)

add_executable(myNode01 src/myNode01.cpp)
target_link_libraries(myNode01 ${catkin_LIBRARIES})
```

add_executable(myNode01 src/myNode01.cpp)
target_link_libraries(myNode01 ${catkin_LIBRARIES})

```
user@user-virtual-machine:~/catkin_ws/src/my_work02/src$ cd ../../..
user@user-virtual-machine:~/catkin_ws$ catkin_make
```

cd ~/catkin_ws
catkin_make 編譯程式

ros node 05

```
user@user-virtual-machine:~/catkin_ws/src/my_work03/src$ rosrun my_work03 myNode01

[ INFO] [1681044576.929375008]: myNode01: hi
```

roscore
rosrun my_work03 myNode01

ros node coding

ROS SERVICE

ros service 01

ros_tutorials_service/src/service_server.cpp https://github.com/Waywrong/ros_course/

https://github.com/Waywrong/ros_course/blob/main/ros_tutorials_service/src/service_server.cpp

Product Solutions Open Source Pricing Search

Waywrong / ros_course Public Notifications Fork 0

Code Issues Pull requests Actions Projects Security Insights

main ros_course / ros_tutorials_service / src / service_server.cpp

Waywrong service Latest commit 65e329a 21 mi

1 contributor

Executable File | 36 lines (28 sloc) | 1.54 KB Raw

```cpp
#include "ros/ros.h"                        // ROS Default Header File
#include "ros_tutorials_service/SrvTutorial.h"// SrvTutorial Service File Header (Automatically created after build)

// The below process is performed when there is a service request
// The service request is declared as 'req', and the service response is declared as 'res'
bool calculation(ros_tutorials_service::SrvTutorial::Request &req,
                 ros_tutorials_service::SrvTutorial::Response &res)
{
  // The service name is 'ros_tutorial_srv' and it will call 'calculation' function upon the service request.
  res.result = req.a + req.b;

  // Displays 'a' and 'b' values used in the service request and
  // the 'result' value corresponding to the service response
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: %ld", (long int)res.result);

  return true;
}

int main(int argc, char **argv)             // Node Main Function
{
  ros::init(argc, argv, "service_server");  // Initializes Node Name
  ros::NodeHandle nh;                       // Node handle declaration

  // Declare service server 'ros_tutorials_service_server'
  // using the 'SrvTutorial' service file in the 'ros_tutorials_service' package.
  // The service name is 'ros_tutorial_srv' and it will call 'calculation' function
```

ros service 02

ser-virtual-machine: ~/catkin_ws/src/my_work03/src

```
l-machine:~$ cd catkin_ws/src/my_work03/src/
l-machine:~/catkin_ws/src/my_work03/src$ gedit myNode01.cpp &
```

myNode01.cpp
~/catkin_ws/src/my_work03/src

Save

```
os.h"
rk03/SrvTutorial.h"

n(my_work03::SrvTutorial::Request &req,
  my_work03::SrvTutorial::Response &res)

req.a + req.b;
uest: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
ding back response: %ld", (long int)res.result);


gc, char **argv)

c, argv, "myNode01");
le n;
erver ros_tutorials_service_server = nh.advertiseService("ros_tutorial_srv", calculation);
o_rate(10);   // 10Hz
ode01: hi");
k())

ce();
leep();
```

```
#include "ros/ros.h"
#include "my_work03/SrvTutorial.h"
bool calculation(my_work03::SrvTutorial::Request &req,
                 my_work03::SrvTutorial::Response &res)
{
  res.result = req.a + req.b;
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: %ld", (long int)res.result);
  return true;
}
int main(int argc, char **argv) {
  ros::init(argc, argv, "myNode01");
  ros::NodeHandle n;
  ros::ServiceServer ros_tutorials_service_server = n.advertiseService("ros_tutorial_srv",
calculation);
  ros::Rate loop_rate(10);   // 10Hz
  ROS_INFO("myNode01: hi");
  while (ros::ok())
  {
    ros::spinOnce();
    loop_rate.sleep();
  }
  return 0;
}
```

ros service 03

https://github.com/Waywrong/ros_course/

ros_tutorials_service/srv/SrvTutorial.srv

Waywrong / **ros_course** Public

Code | Issues | Pull requests | Actions | Projects | Security

main | ros_course / ros_tutorials_service / srv / **SrvTutorial.srv**

Waywrong service

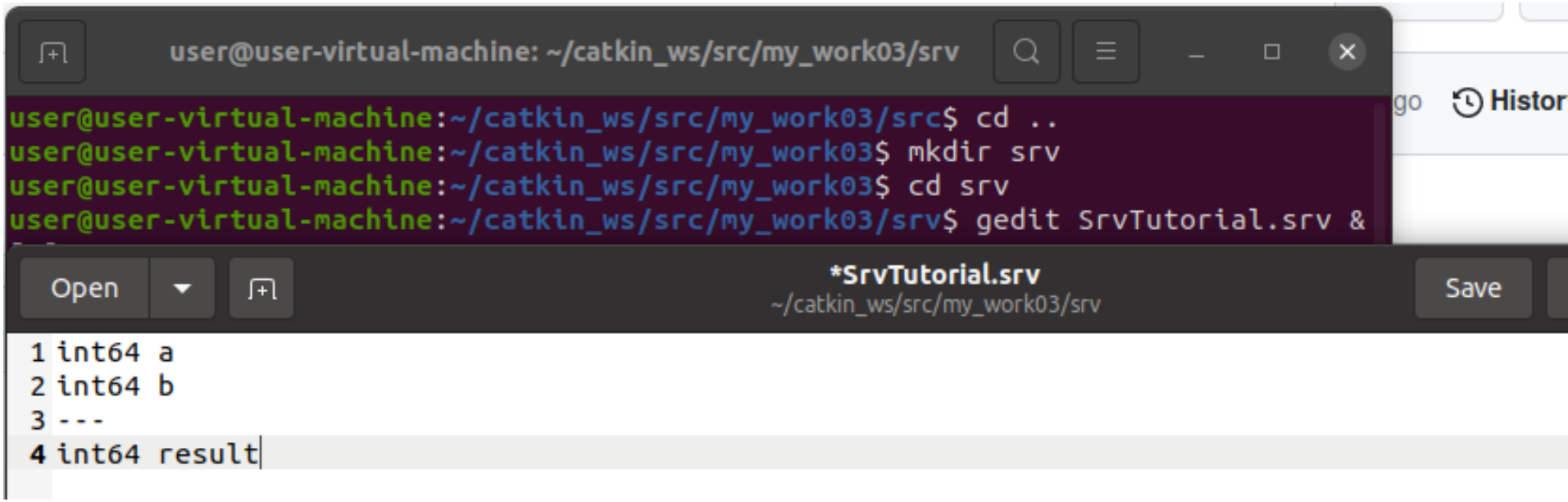
1 contributor

Executable File | 4 lines (4 sloc) | 33 Bytes

```
int64 a
int64 b
---
int64 result
```

ros service 04

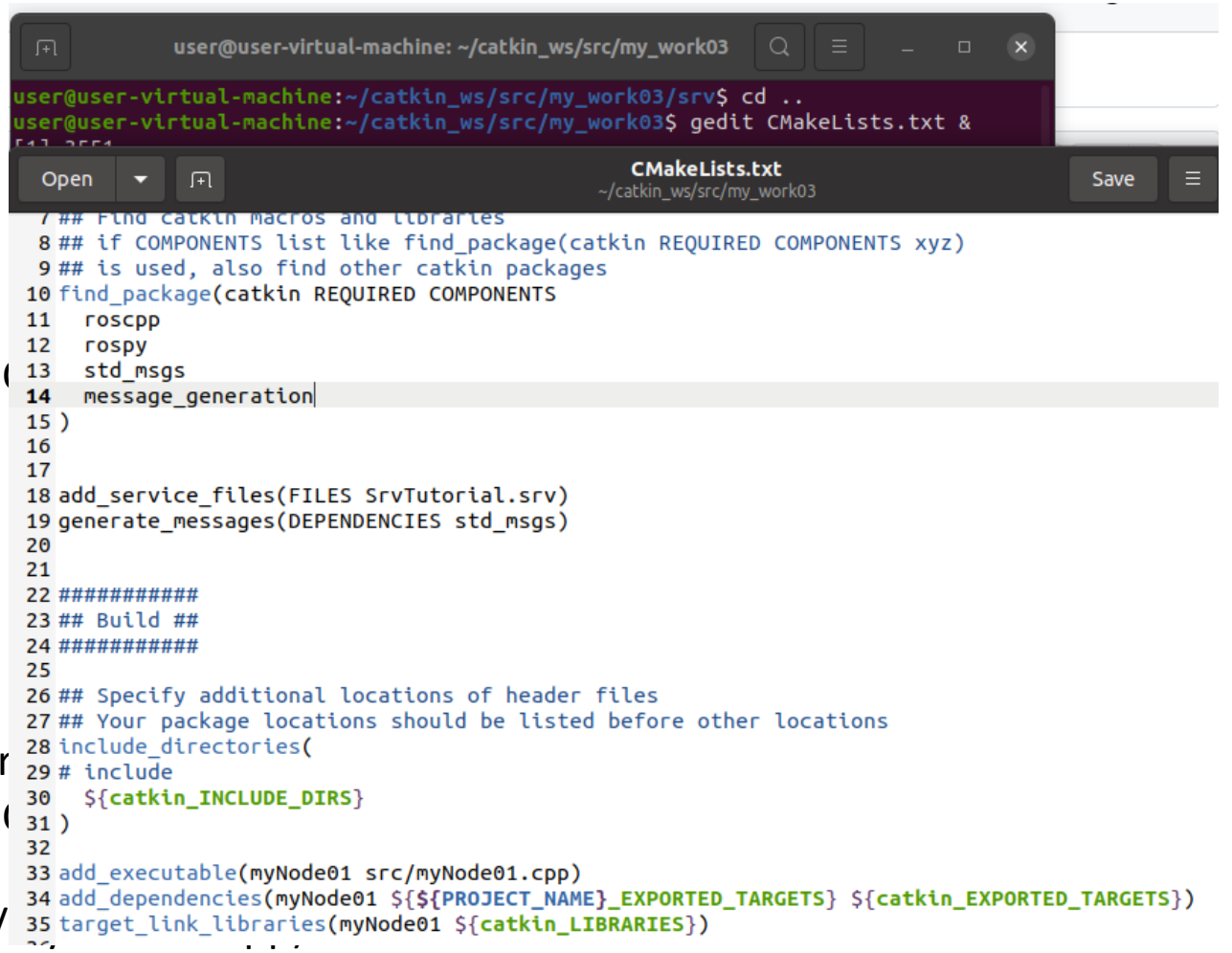
cd ..
mkdir srv
cd srv
gedit SrvTutorial.srv &



int64 a
int64 b
---
int64 result

ros service 05

```
cd ..
gedit CMakeLists.txt &
```




```
find_package(catkin REQUIRED (
  roscpp
  rospy
  std_msgs
  message_generation
)

add_service_files(FILES SrvTutor
generate_messages(DEPENDEN(

add_executable(myNode01 src/
add_dependencies(myNode01
${${PROJECT_NAME}_EXPORTED_TARGETS}
${catkin_EXPORTED_TARGETS})
target_link_libraries(myNode01
${catkin_LIBRARIES})
```

```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  message_generation
)

add_service_files(FILES SrvTutorial.srv)
generate_messages(DEPENDENCIES std_msgs)

add_executable(myNode01 src/myNode01.cpp)
add_dependencies(myNode01 ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
target_link_libraries(myNode01 ${catkin_LIBRARIES})
```

表示這個程式編譯時有先後相依問題(SrvTutorial.h)，無此行則需多編譯數次才可成功

ros service 06

```
user@user-virtual-machine:~/catkin_ws$ catkin_make
Base path: /home/user/catkin_ws
Source space: /home/user/catkin_ws/src
Build space: /home/user/catkin_ws/build
Devel space: /home/user/catkin_ws/devel
Install space: /home/user/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/user/catkin_ws/b
uild"
```

cd ~/catkin_ws
catkin_make 編譯程式

ros service 07

```
user@user-virtual-machine:~/catkin_ws$ rosrun my_work03 myNode01
[ INFO] [1681050254.623201291]: myNode01: hi
[ INFO] [1681050310.023610107]: request: x=2, y=3
[ INFO] [1681050310.023710495]: sending back response: 5
```

roscore
rosrun my_work03 myNode01

```
roscore http://user-v...    user@user-virtual-m...    user@user-virtual-m...
user@user-virtual-machine:~/catkin_ws$ rosservice list
/myNode01/get_loggers
/myNode01/set_logger_level
/ros_tutorial_srv
/rosout/get_loggers
/rosout/set_logger_level
user@user-virtual-machine:~/catkin_ws$ rosservice call /ros_tutorial_srv "a: 2
b: 3"
result: 5
user@user-virtual-machine:~/catkin_ws$
```

rosservice list
rosservice call /ros_tutorial_srv 2 3

作業3

- 於本周相同專案內(my_work03)增加一個service client，並上傳相關檔案

- 參考4/11上課內容，"ROS-Class-6.pdf“
- ros_tutorials_service/src/service_client.cpp，寫一個會送出服務請求(service request)的專案(my_hw03)
- 計分部分包含
  - 1. 完整性
  - 2.修改rosnode 名稱
  - 3. 紀錄實驗過程於word檔，紀錄所下的命令與回應，可多利用截圖(圖文並茂加分)
    - rosnode list
    - rosservice list
    - Rosservice info

- 上傳作業包含 (4/25 前上傳):
- 1. CMakeLists.txt
- 2. package.xml
- 3. 修改過的cpp檔
- 4. 實驗紀錄word檔

https://github.com/Waywrong/ros_course/blob/main/ros_tutorials_service/src/service_client.cpp

Code | Issues | Pull requests | Actions | Projects | Security | Insights

main | ros_course / ros_tutorials_service / src / **service_client.cpp** | Go to file | ···

Waywrong service | Latest commit 65e329a 1 hour ago | History

1 contributor

Executable File | 42 lines (35 sloc) | 1.71 KB | Raw | Blame

```cpp
#include "ros/ros.h"                        // ROS Default Header File
#include "ros_tutorials_service/SrvTutorial.h"// SrvTutorial Service File Header (Automatically created after build)
#include <cstdlib>                          // Library for using the "atoll" function

int main(int argc, char **argv)             // Node Main Function
{
  ros::init(argc, argv, "service_client");    // Initializes Node Name

  if (argc != 3)  // Input value error handling
  {
    ROS_INFO("cmd : rosrun ros_tutorials_service service_client arg0 arg1");
    ROS_INFO("arg0: double number, arg1: double number");
    return 1;
  }

  ros::NodeHandle nh;       // Node handle declaration for communication with ROS system

  // Declares service client 'ros_tutorials_service_client'
  // using the 'SrvTutorial' service file in the 'ros_tutorials_service' package.
  // The service name is 'ros_tutorial_srv'
  ros::ServiceClient ros_tutorials_service_client = nh.serviceClient<ros_tutorials_service::SrvTutorial>("ros_tutorial_srv");

  // Declares the 'srv' service that uses the 'SrvTutorial' service file
  ros_tutorials_service::SrvTutorial srv;

  // Parameters entered when the node is executed as a service request value are stored at 'a' and 'b'
  srv.request.a = atoll(argv[1]);
  srv.request.b = atoll(argv[2]);

  // Request the service. If the request is accepted, display the response value
  if (ros_tutorials_service_client.call(srv))
  {
    ROS_INFO("send srv, srv.Request.a and b: %ld, %ld", (long int)srv.request.a, (long int)srv.request.b);
    ROS_INFO("receive srv, srv.Response.result: %ld", (long int)srv.response.result);
  }
  else
  {
    ROS_ERROR("Failed to call service ros_tutorial_srv");
    return 1;
  }
  return 0;
}
```