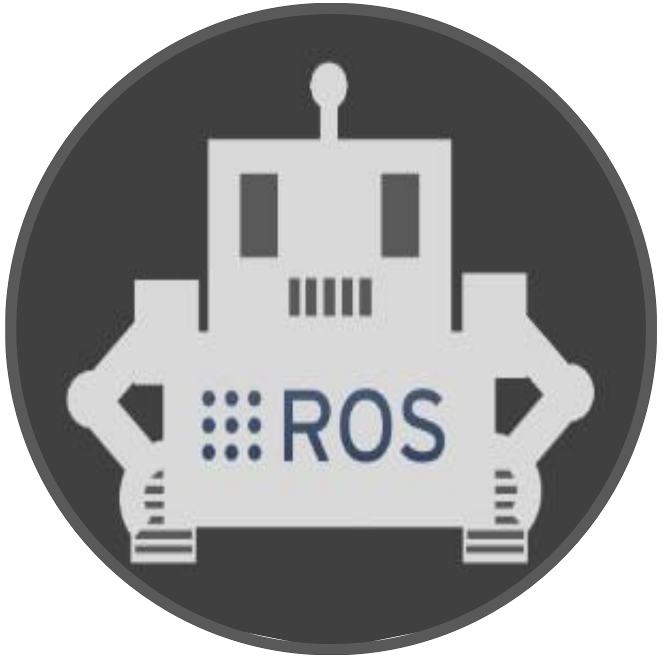
ROS 기초 강의

Chapter 6. ROS Service

구선생 로보틱스



Update 20240620

강의 자료 다운로드

ROS 기초 강의 강의노트

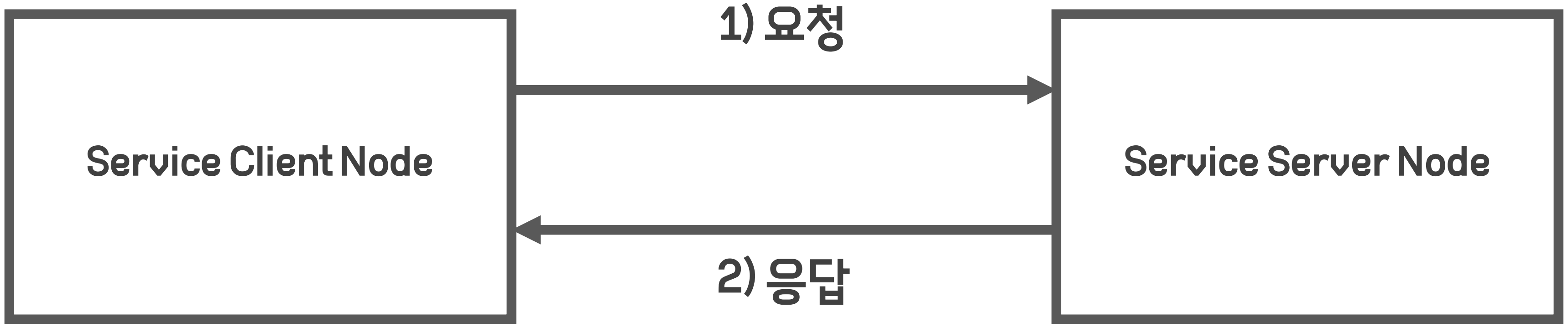
https://drive.google.com/drive/folders/1rRwS2j98HNyj5Is_yVXEGj30ILvMPtrz?usp=drive_link

1. ROS Service 기초
2. Service 패키지 생성
3. Service Service Node
4. Service Client Node

ROS Service 기초

ROS Service란?

동영상 강의 - ROS Service 기초
https://youtu.be/c5wA7ze88JE?si=wizWjquVPQ0exp7V



일회성 메시지 통신

로봇의 일회성 동작 명령을 위해서 주로 사용한다.

ROS Service 기초

ROS Service 명령어

- Service 리스트 보기

```
$ rosservice list
```

- Service 요청

```
$ rosservice call <서비스_이름> <서비스_양식>
```

- Service 상세 정보 확인

```
$ rosservice info <서비스_이름>
```

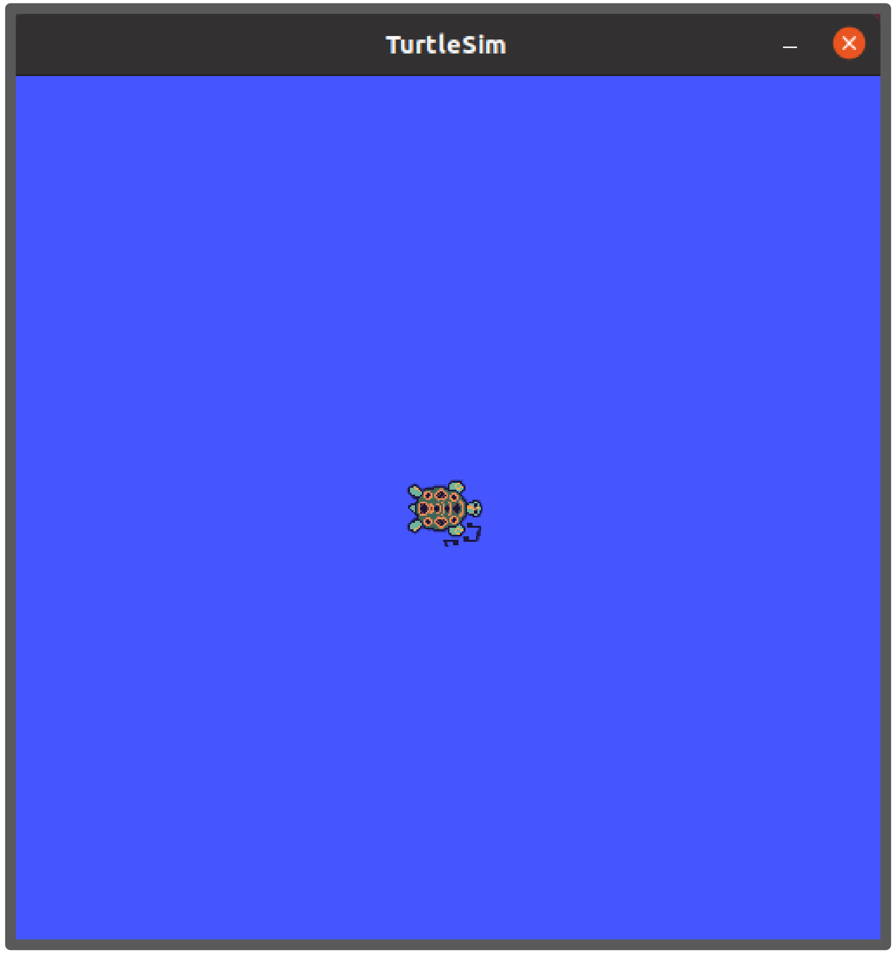
기타 명령어는 아래 위키 참고

http://wiki.ros.org/rosservice

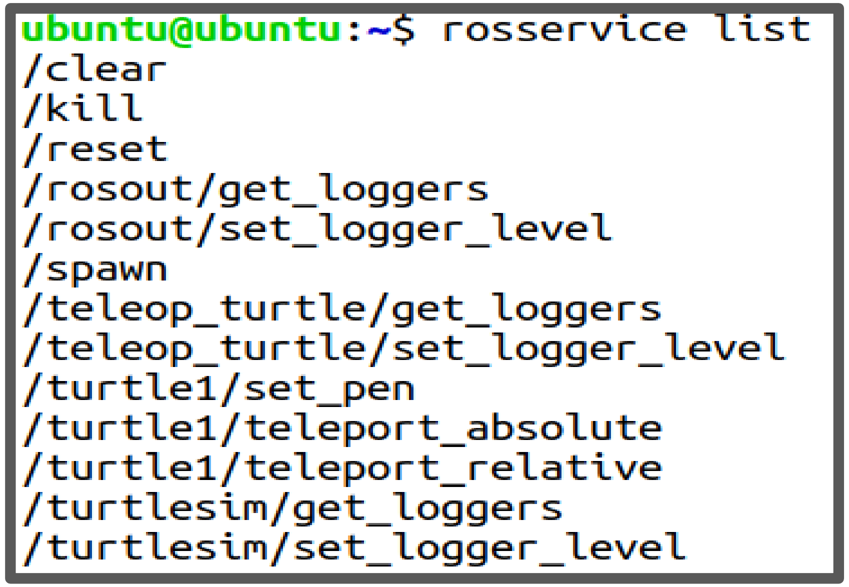
ROS Service 기초

ROS Service 명령어

TurtleSim Node의 Service

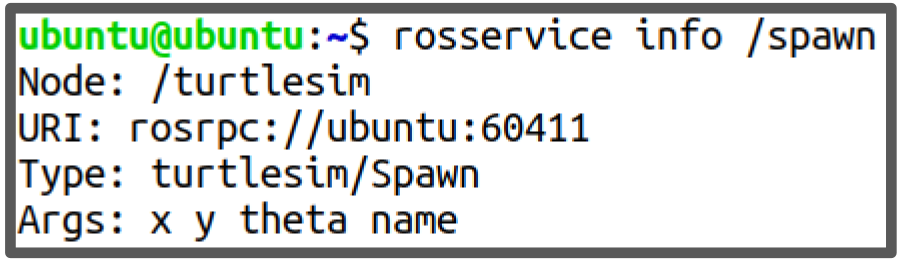


- rosservice list의 결과



```
ubuntu@ubuntu:~$ rosservice list
/clear
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/spawn
/teleop_turtle/get_loggers
/teleop_turtle/set_logger_level
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/get_loggers
/turtlesim/set_logger_level
```

- rosservice info의 결과

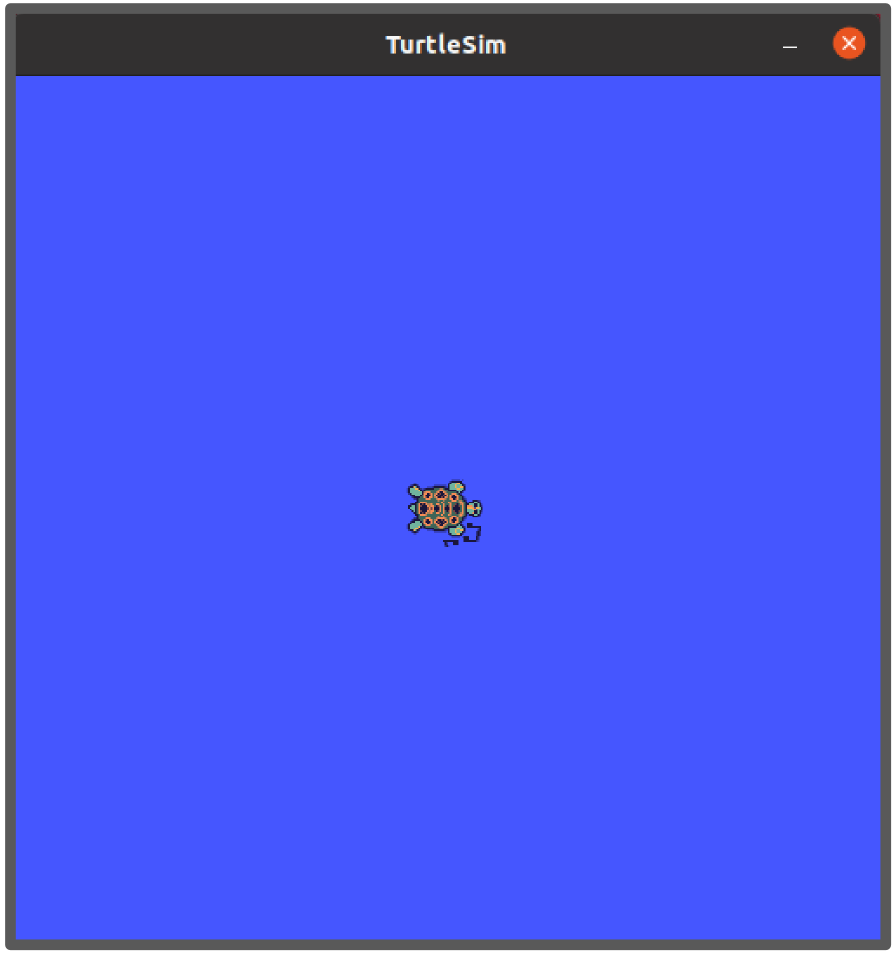


```
ubuntu@ubuntu:~$ rosservice info /spawn
Node: /turtlesim
URI: rosrpc://ubuntu:60411
Type: turtlesim/Spawn
Args: x y theta name
```

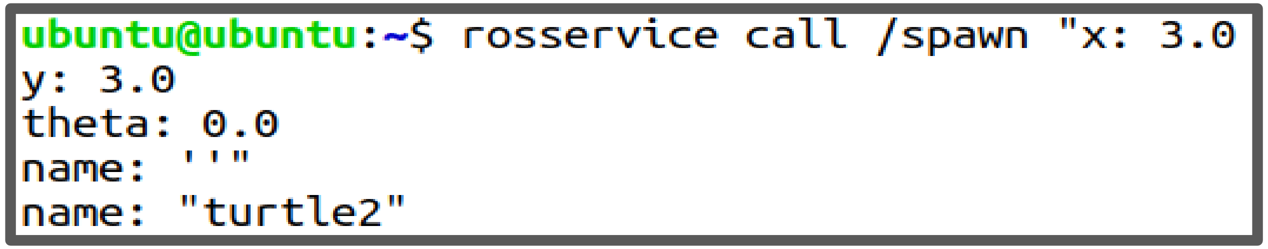
ROS Service 기초

ROS Service 명령어

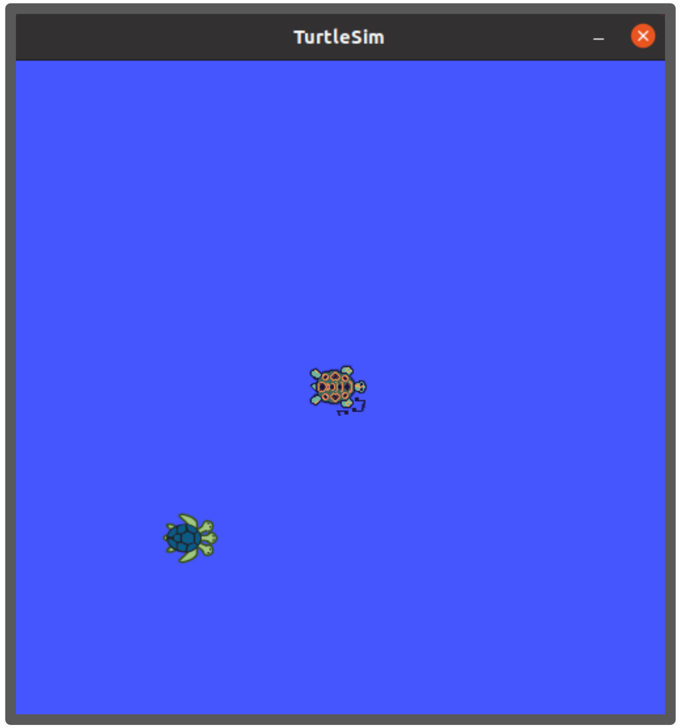
TurtleSim Node의 Service



- rosservice call의 결과



```
ubuntu@ubuntu:~$ rosservice call /spawn "x: 3.0
y: 3.0
theta: 0.0
name: ''"
name: "turtle2"
```



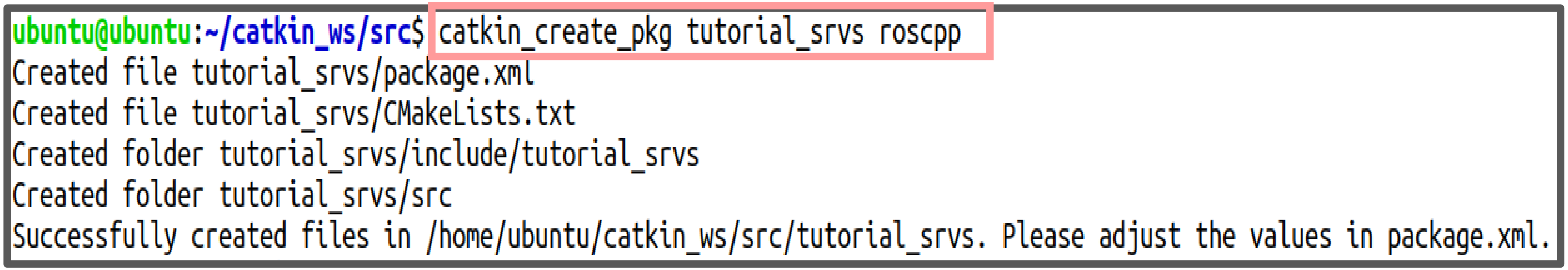
1. ROS Service 기초
2. **Service 패키지 생성**
3. Service Server Node
4. Service Client Node

Service 패키지 생성

동영상 강의 - Service Server Node 작성
https://youtu.be/6-J-zFJiCMY?si=fbm3QYm4pw9wfm7w

1) 패키지 생성

```
$ catkin_create_pkg tutorial_srvs roscpp
```



```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg tutorial_srvs roscpp
Created file tutorial_srvs/package.xml
Created file tutorial_srvs/CMakeLists.txt
Created folder tutorial_srvs/include/tutorial_srvs
Created folder tutorial_srvs/src
Successfully created files in /home/ubuntu/catkin_ws/src/tutorial_srvs. Please adjust the values in package.xml.
```

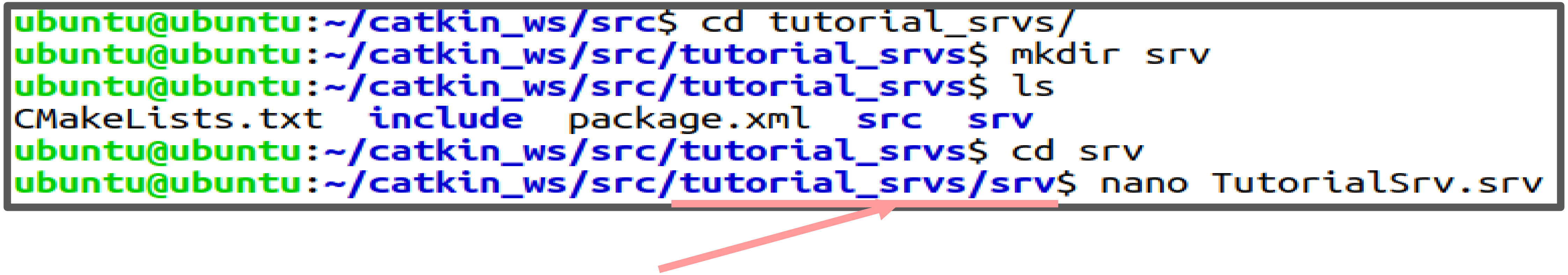
∵ 서비스 패키지는 모듈화를 위해 독립된 패키지로 생성하는 것을 권장

소스코드 - tutorial_srvs
https://github.com/PigeonSensei/pigeon_ros_tutorial/tree/master/others/tutorial_srvs

Service 패키지 생성

2) 서비스 파일 생성

```
$ nano TutorialMsg.msg
```

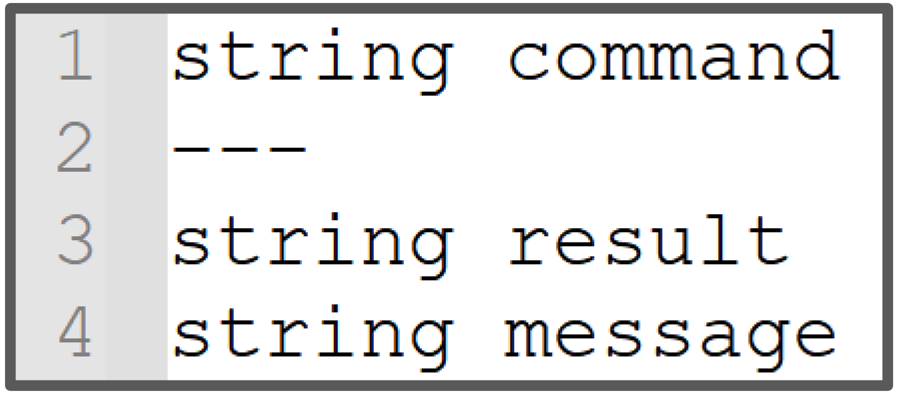


```
ubuntu@ubuntu:~/catkin_ws/src$ cd tutorial_srvs/
ubuntu@ubuntu:~/catkin_ws/src/tutorial_srvs$ mkdir srv
ubuntu@ubuntu:~/catkin_ws/src/tutorial_srvs$ ls
CMakeLists.txt  include  package.xml  src  srv
ubuntu@ubuntu:~/catkin_ws/src/tutorial_srvs$ cd srv
ubuntu@ubuntu:~/catkin_ws/src/tutorial_srvs/srv$ nano TutorialSrv.srv
```

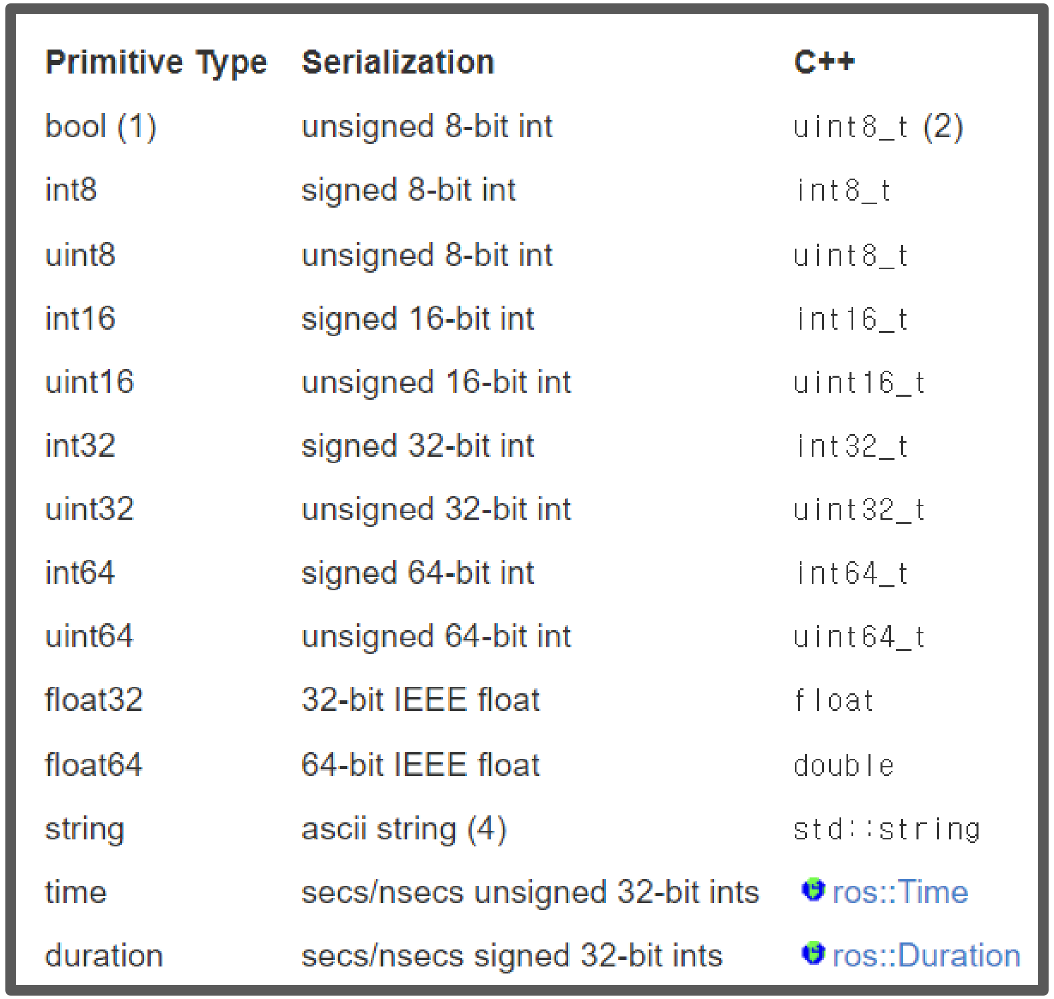
∵ 서비스 파일은 패키지의 srv 경로에서 생성되어야 한다.
srv 경로가 없다면 생성한다.

Service 패키지 생성

3) 서비스 파일 작성



```
string command
---
string result
string message
```



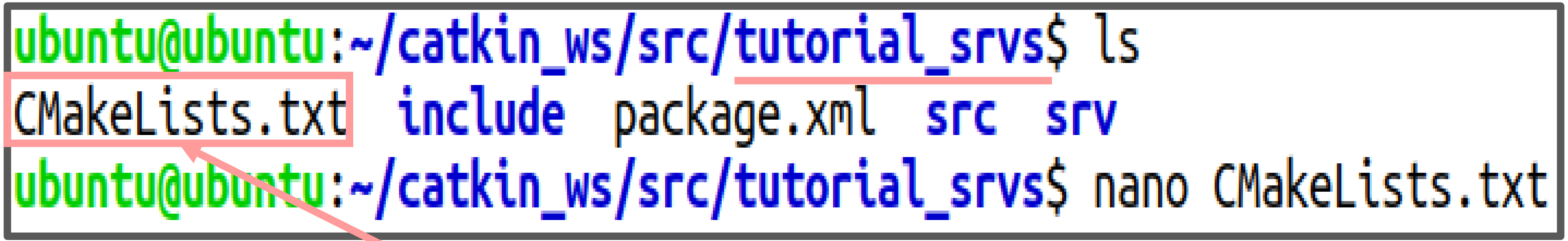
Primitive Type	Serialization	C++
bool (1)	unsigned 8-bit int	uint8_t (2)
int8	signed 8-bit int	int8_t
uint8	unsigned 8-bit int	uint8_t
int16	signed 16-bit int	int16_t
uint16	unsigned 16-bit int	uint16_t
int32	signed 32-bit int	int32_t
uint32	unsigned 32-bit int	uint32_t
int64	signed 64-bit int	int64_t
uint64	unsigned 64-bit int	uint64_t
float32	32-bit IEEE float	float
float64	64-bit IEEE float	double
string	ascii string (4)	std::string
time	secs/nsecs unsigned 32-bit ints	ros::Time
duration	secs/nsecs signed 32-bit ints	ros::Duration

서비스 파일에서 사용 가능한 타입은
msg 파일과 동일하다.

Service 패키지 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```

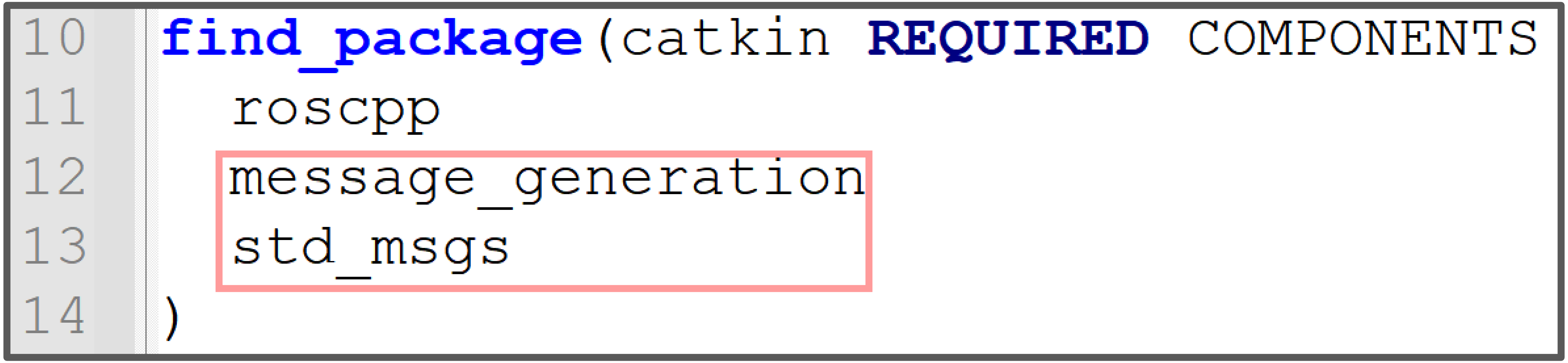


```
ubuntu@ubuntu:~/catkin_ws/src/tutorial_srvs$ ls
CMakeLists.txt  include  package.xml  src  srv
ubuntu@ubuntu:~/catkin_ws/src/tutorial_srvs$ nano CMakeLists.txt
```

∵ CMakeLists.txt는 패키지 경로에 있다.

Service 패키지 생성

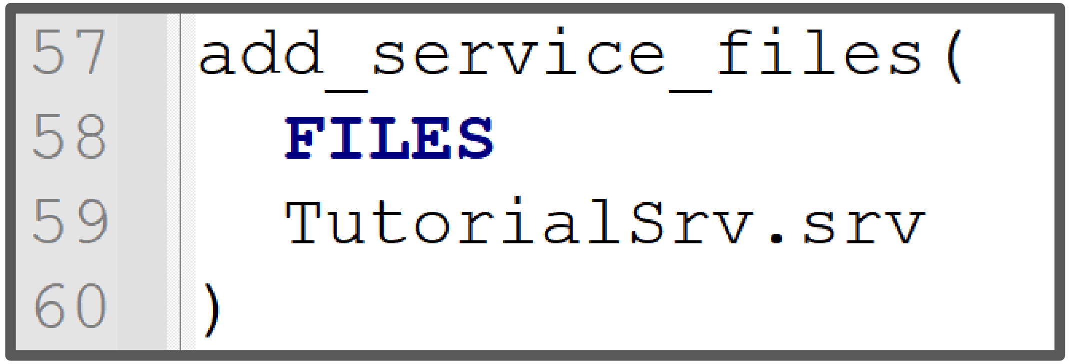
5) CMakeLists.txt의 find_package 내용 추가



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

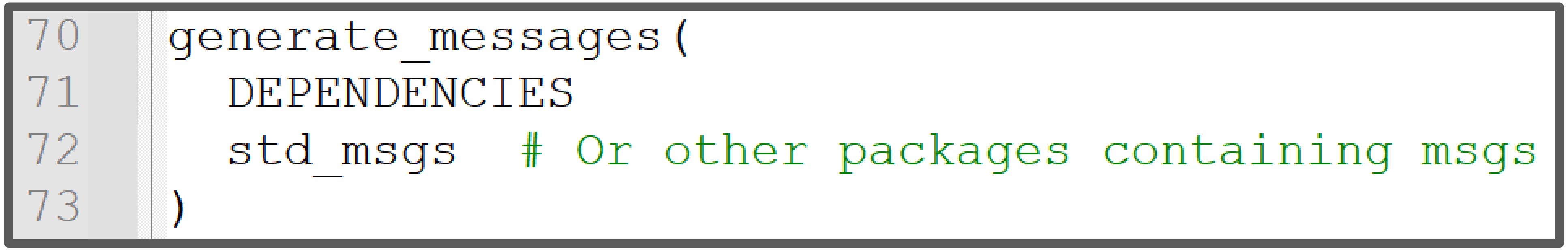
Service 패키지 생성

6) CMakeLists.txt add_service_files, generate_messages 추가



```
add_service_files(
  FILES
  TutorialSrv.srv
)
```

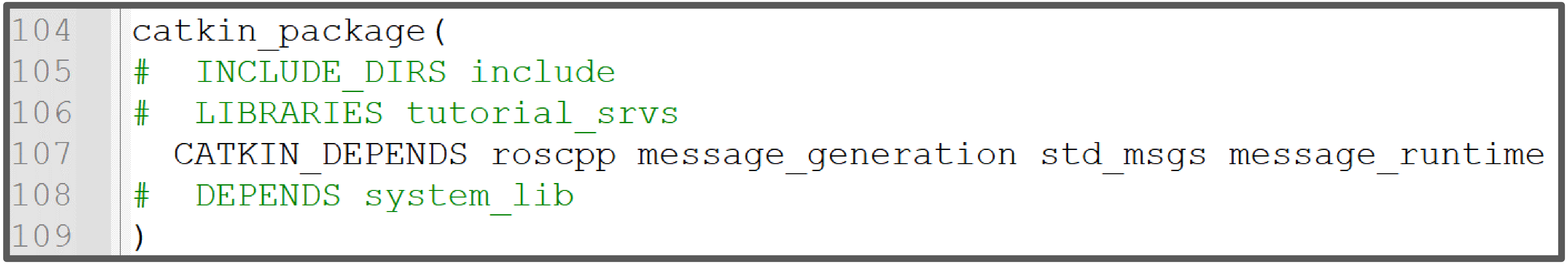
⋮



```
generate_messages(
  DEPENDENCIES
  std_msgs  # Or other packages containing msgs
)
```

Service 패키지 생성

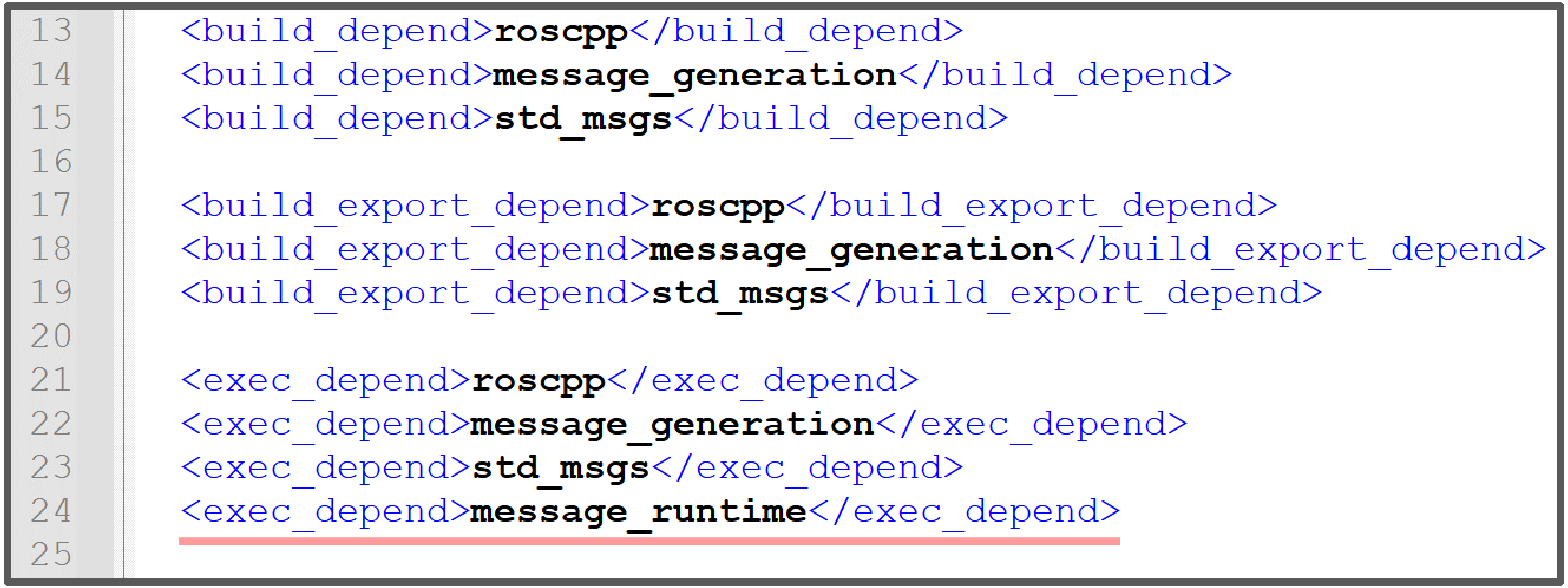
7) CMakeLists.txt catkin_package의 CATKIN_DEPENDS 수정



```
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES tutorial_srvs
  CATKIN_DEPENDS roscpp message_generation std_msgs message_runtime
#  DEPENDS system_lib
)
```

Service 패키지 생성

8) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

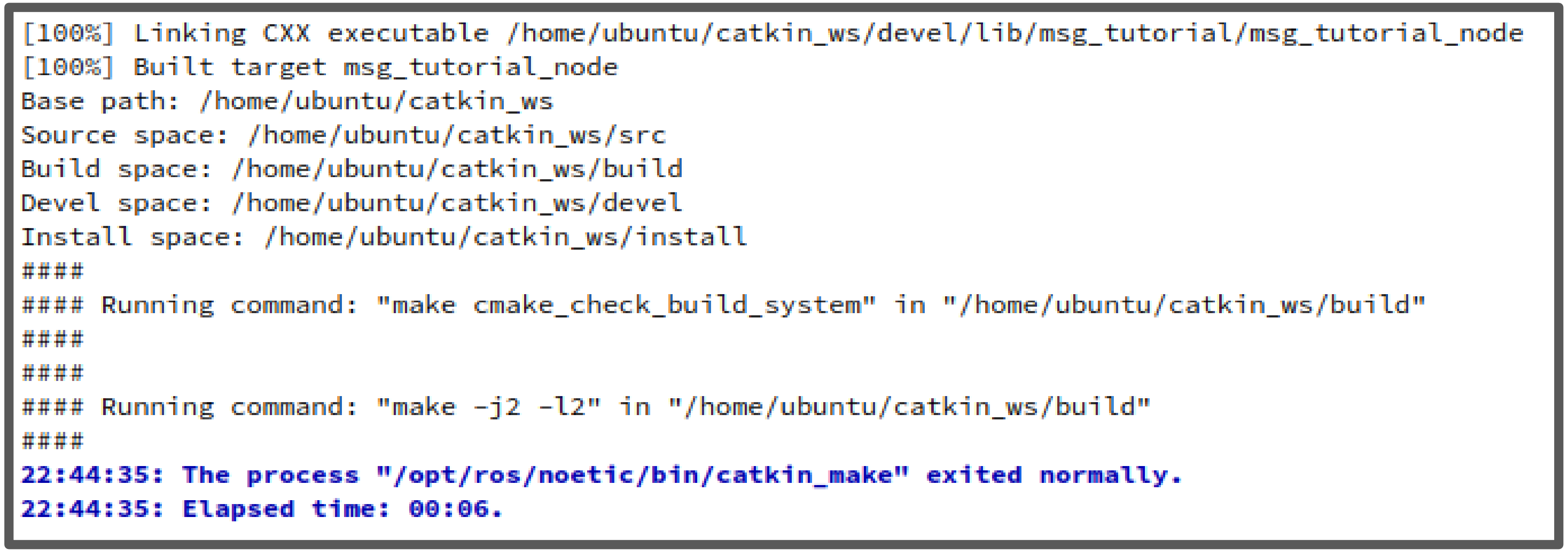


```
    <build_depend>roscpp</build_depend>
    <build_depend>message_generation</build_depend>
    <build_depend>std_msgs</build_depend>

    <build_export_depend>roscpp</build_export_depend>
    <build_export_depend>message_generation</build_export_depend>
    <build_export_depend>std_msgs</build_export_depend>

    <exec_depend>roscpp</exec_depend>
    <exec_depend>message_generation</exec_depend>
    <exec_depend>std_msgs</exec_depend>
    <exec_depend>message_runtime</exec_depend>

```

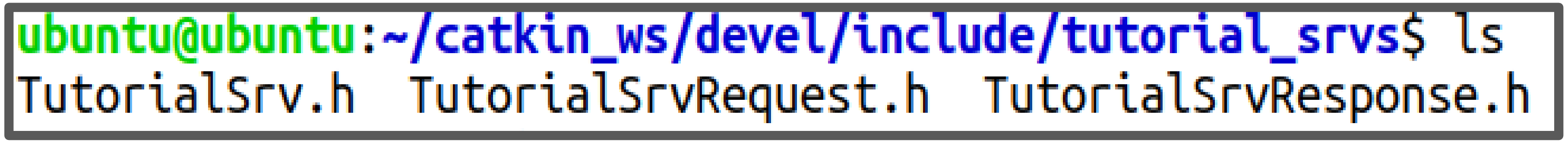
∵ message_runtime 추가 작성

Service 패키지 생성

9) 컴파일 후 라이브러리 생성 확인



```
[100%] Linking CXX executable /home/ubuntu/catkin_ws/devel/lib/msg_tutorial/msg_tutorial_node
[100%] Built target msg_tutorial_node
Base path: /home/ubuntu/catkin_ws
Source space: /home/ubuntu/catkin_ws/src
Build space: /home/ubuntu/catkin_ws/build
Devel space: /home/ubuntu/catkin_ws/devel
Install space: /home/ubuntu/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/ubuntu/catkin_ws/build"
####
####
#### Running command: "make -j2 -l2" in "/home/ubuntu/catkin_ws/build"
####
22:44:35: The process "/opt/ros/noetic/bin/catkin_make" exited normally.
22:44:35: Elapsed time: 00:06.
```



```
ubuntu@ubuntu:~/catkin_ws/devel/include/tutorial_srvs$ ls
TutorialSrv.h  TutorialSrvRequest.h  TutorialSrvResponse.h
```

컴파일이 완료되고 해당 위치에 서비스 파일 이름의 헤더파일이 생성 되어야 사용가능 하다

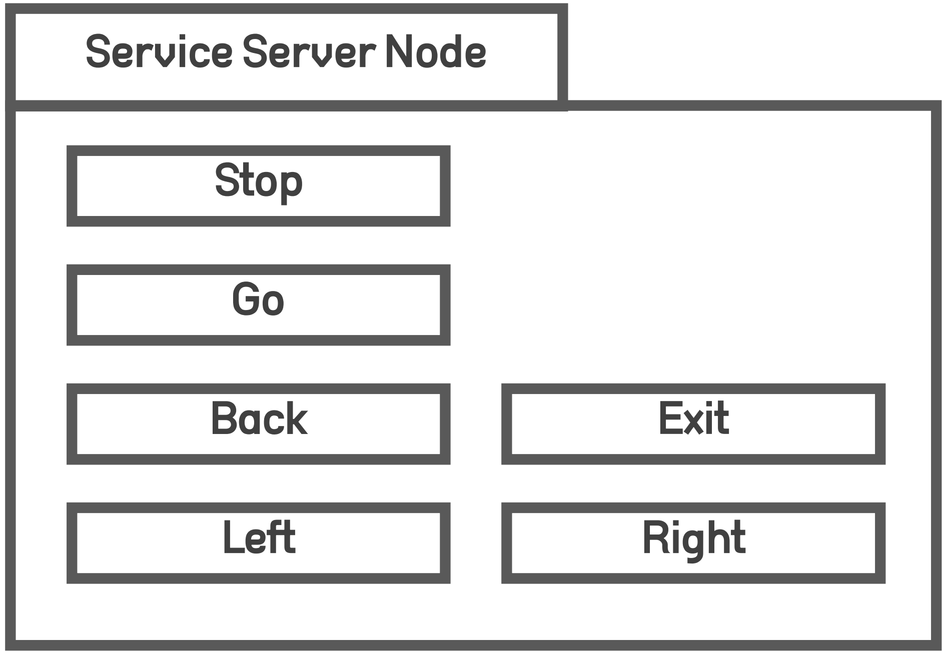
1. ROS Service 기초
2. Service 패키지 생성
3. **Service Server Node**
4. Service Client Node

Service Server Node

Service Server Node란?

동영상 강의 - Service Server Node 작성
https://youtu.be/6-J-zFJiCMY?si=fbm3QYm4pw9wfm7w

서비스 통신에서 서비스 제공을 담당



Service Server Node

Stop

Go

Back

Exit

Left

Right

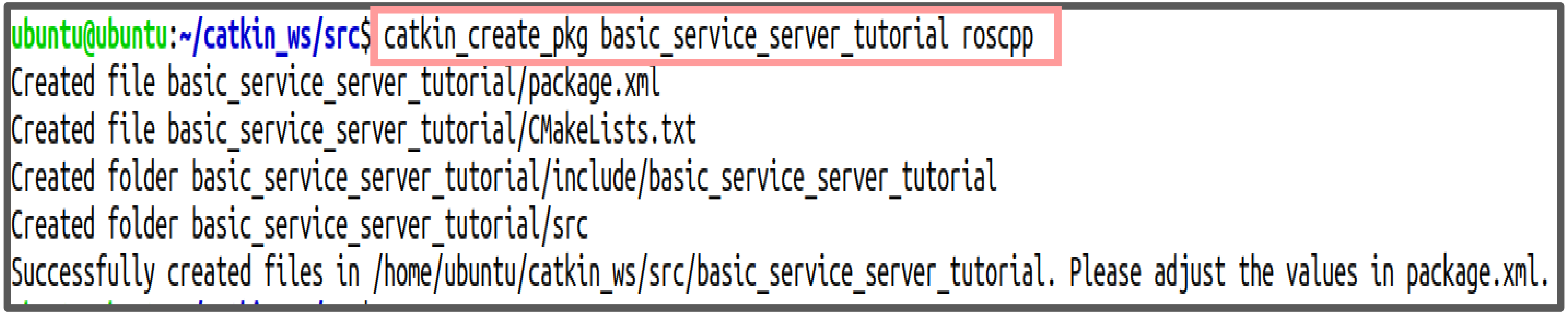
로봇의 리모콘을 만드는 역할이라고 생각하면 이해하기 쉽다.

Service Server Node

Service Server Node 생성

1) 패키지 생성

```
$ catkin_create_pkg basic_service_server_tutorial roscpp
```



```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg basic_service_server_tutorial roscpp
Created file basic_service_server_tutorial/package.xml
Created file basic_service_server_tutorial/CMakeLists.txt
Created folder basic_service_server_tutorial/include/basic_service_server_tutorial
Created folder basic_service_server_tutorial/src
Successfully created files in /home/ubuntu/catkin_ws/src/basic_service_server_tutorial. Please adjust the values in package.xml.
```

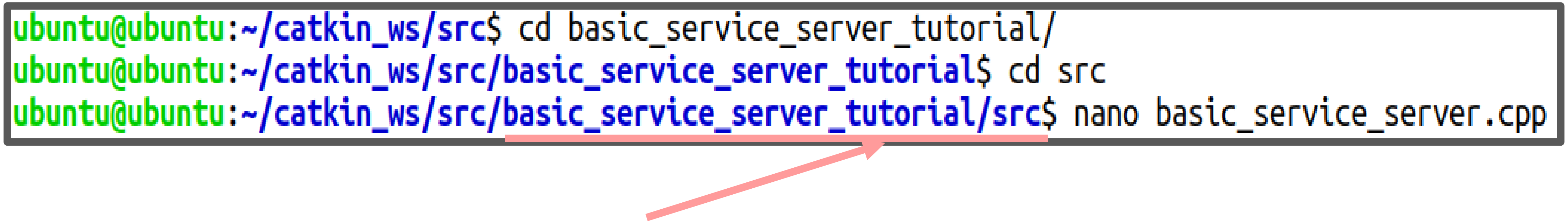
소스코드 – basic_service_server_tutorial
https://github.com/PigeonSensei/pigeon_ros_tutorial/tree/master/basic/basic_service_server_tutorial

Service Server Node

Service Server Node 생성

2) 소스코드 파일 생성

```
$ nano basic_service_server.cpp
```



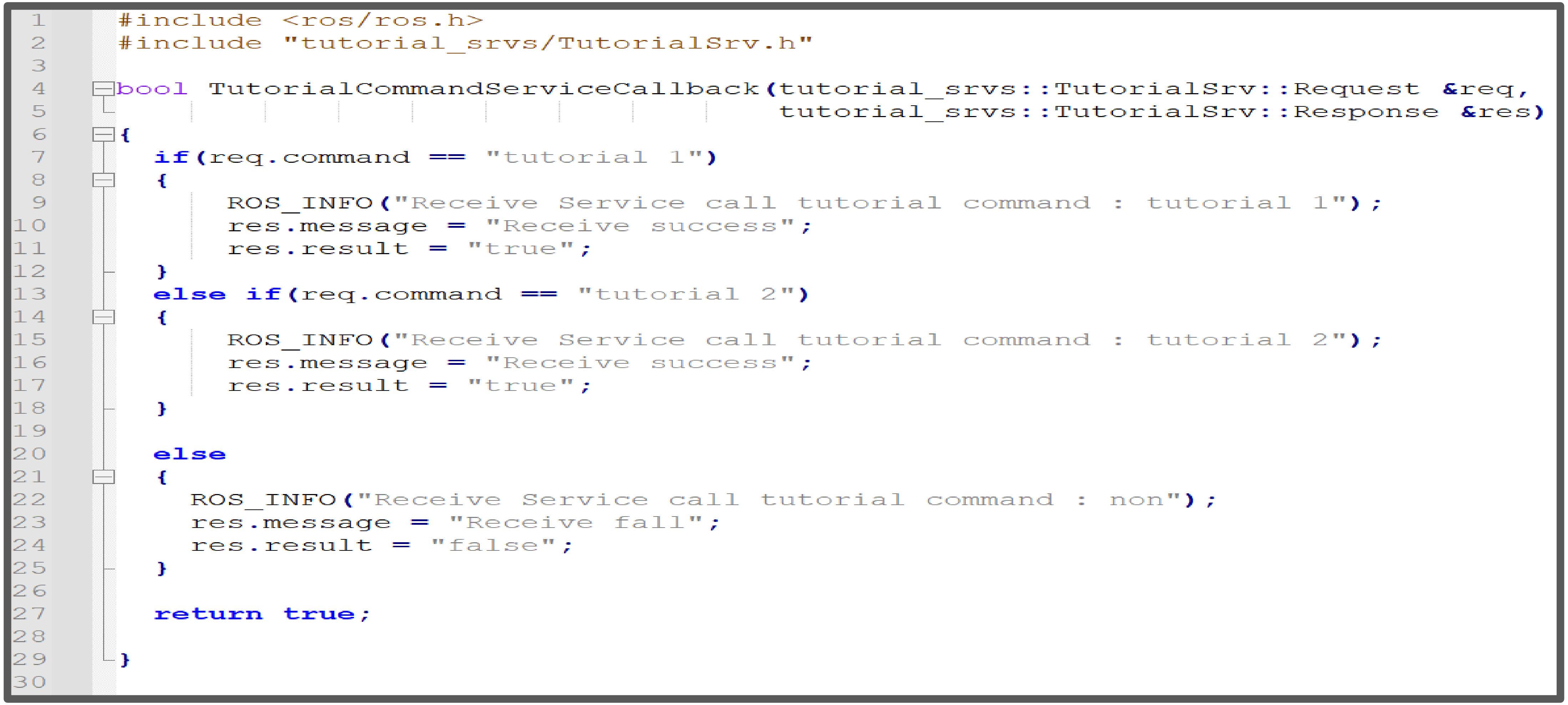
```
ubuntu@ubuntu:~/catkin_ws/src$ cd basic_service_server_tutorial/
ubuntu@ubuntu:~/catkin_ws/src/basic_service_server_tutorial$ cd src
ubuntu@ubuntu:~/catkin_ws/src/basic_service_server_tutorial/src$ nano basic_service_server.cpp
```

∵ 소스코드는 패키지의 src 경로에서 생성되어야 한다.

Service Server Node

Service Server Node 생성

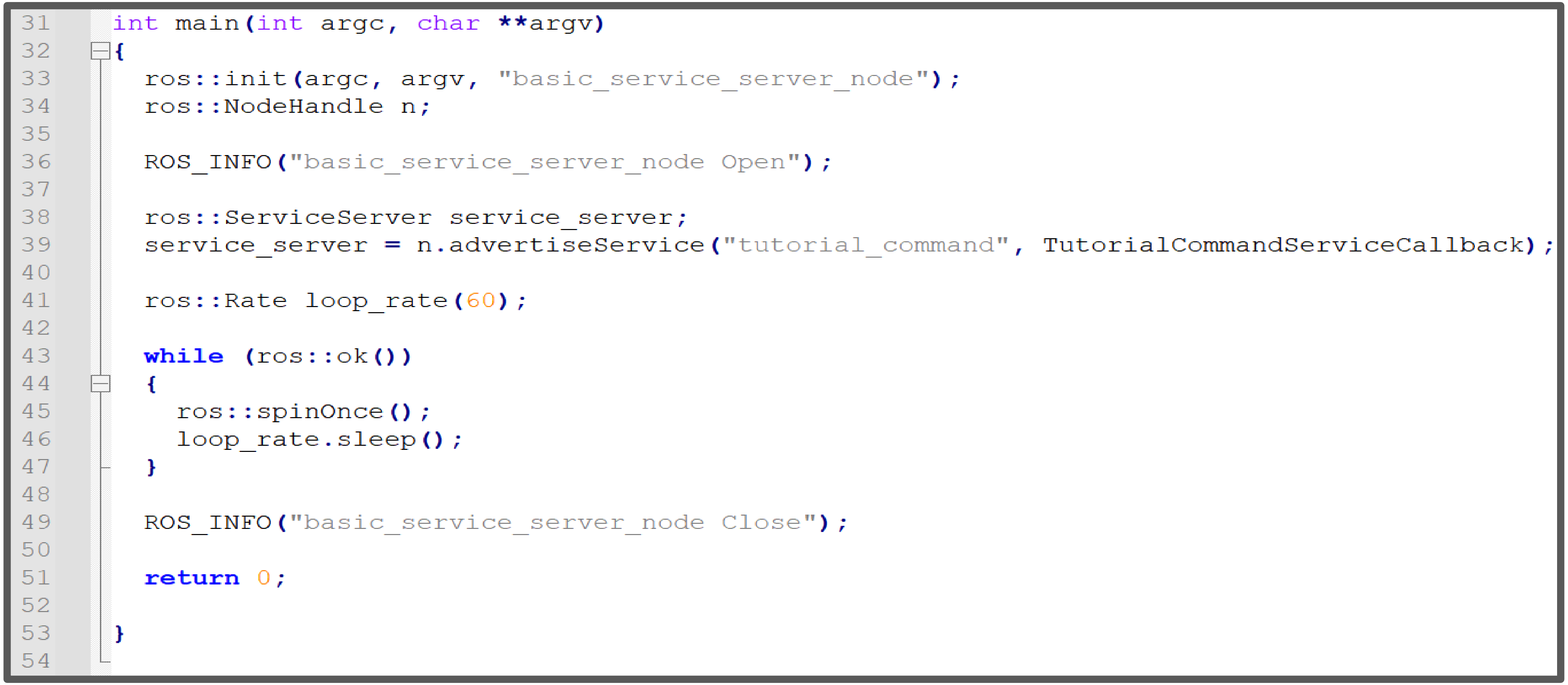
3) 소스코드 작성



```cpp
#include <ros/ros.h>
#include "tutorial_srvs/TutorialSrv.h"

bool TutorialCommandServiceCallback(tutorial_srvs::TutorialSrv::Request &req,
                                    tutorial_srvs::TutorialSrv::Response &res)
{
  if(req.command == "tutorial 1")
  {
      ROS_INFO("Receive Service call tutorial command : tutorial 1");
      res.message = "Receive success";
      res.result = "true";
  }
  else if(req.command == "tutorial 2")
  {
      ROS_INFO("Receive Service call tutorial command : tutorial 2");
      res.message = "Receive success";
      res.result = "true";
  }

  else
  {
    ROS_INFO("Receive Service call tutorial command : non");
    res.message = "Receive fall";
    res.result = "false";
  }

  return true;

}

```

Service Server Node

Service Server Node 생성



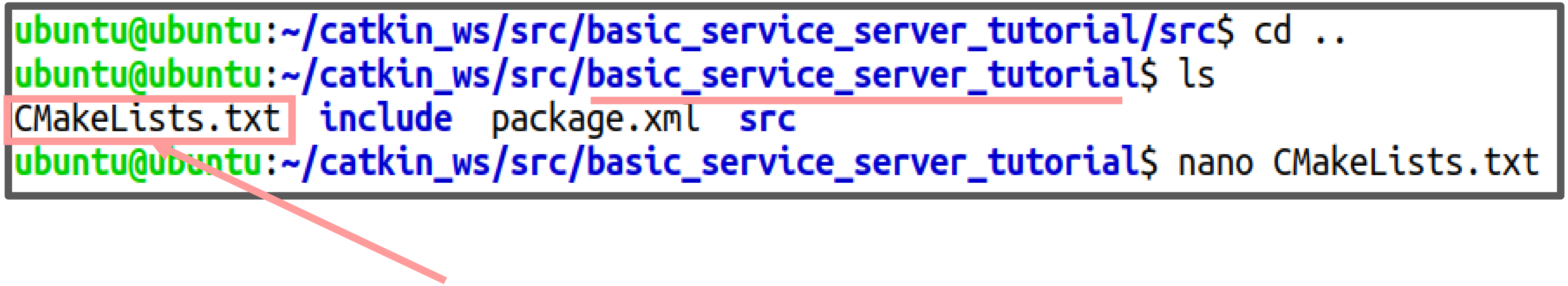
```cpp
int main(int argc, char **argv)
{
  ros::init(argc, argv, "basic_service_server_node");
  ros::NodeHandle n;

  ROS_INFO("basic_service_server_node Open");

  ros::ServiceServer service_server;
  service_server = n.advertiseService("tutorial_command", TutorialCommandServiceCallback);

  ros::Rate loop_rate(60);

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

  ROS_INFO("basic_service_server_node Close");

  return 0;

}
```

Service Server Node

Service Server Node 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```



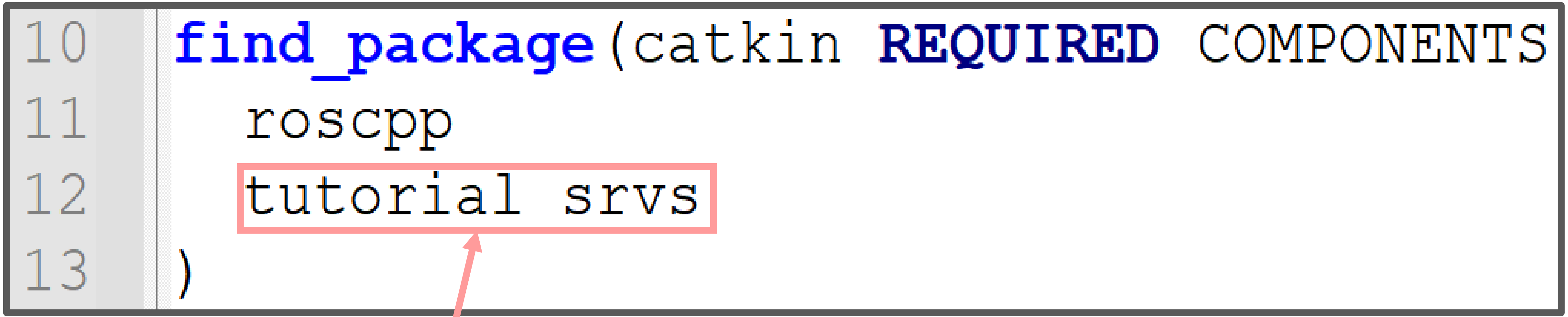
```
ubuntu@ubuntu:~/catkin_ws/src/basic_service_server_tutorial/src$ cd ..
ubuntu@ubuntu:~/catkin_ws/src/basic_service_server_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_service_server_tutorial$ nano CMakeLists.txt
```

∵ CMakeLists.txt는 패키지 경로에 있다.

Service Server Node

Service Server Node 생성

5) CMakeLists.txt의 find_package 내용 추가



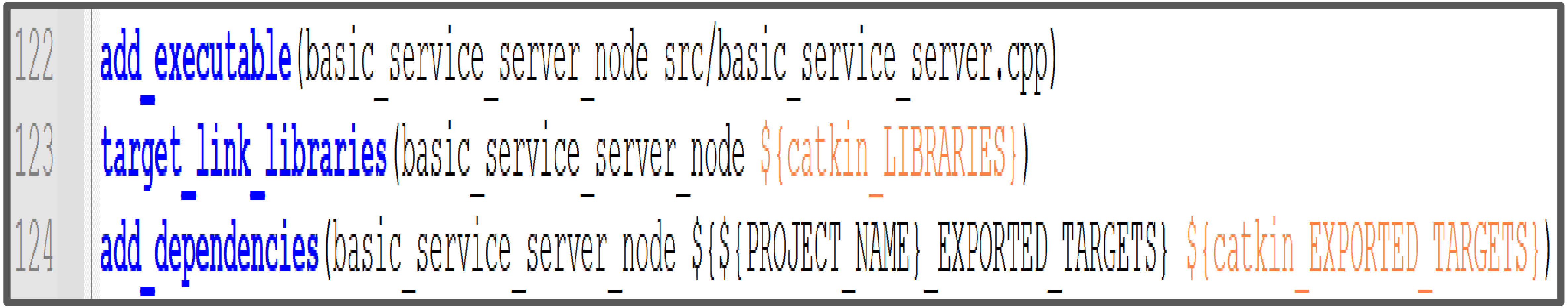
```
find_package(catkin REQUIRED COMPONENTS
   roscpp
   tutorial srvs
)
```

∵ tutorial_srvs는 서비스 패키지 이름이다.

Service Server Node

Service Server Node 생성

6) CMakeLists.txt add_executable, target_link_librariese, add_dependencies 추가

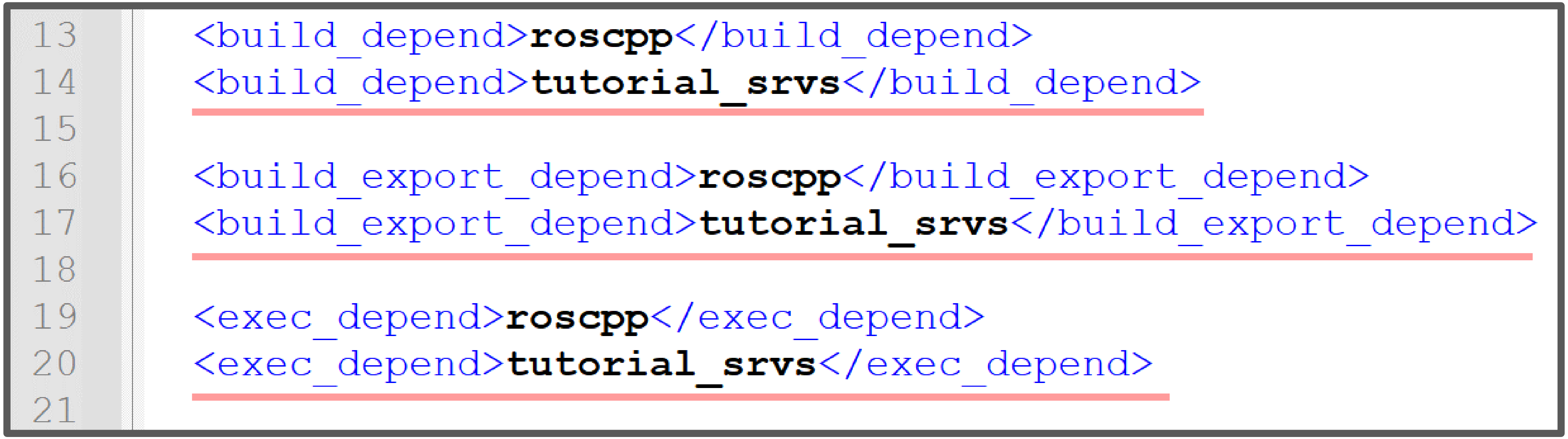


```
add_executable(basic_service_server_node src/basic_service_server.cpp)
target_link_libraries(basic_service_server_node ${catkin_LIBRARIES})
add_dependencies(basic_service_server_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
```

Service Server Node

Service Server Node 생성

7) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

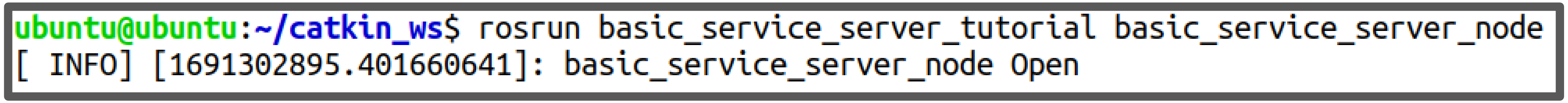


```
  <build_depend>roscpp</build_depend>
  <build_depend>tutorial_srvs</build_depend>

  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>tutorial_srvs</build_export_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>tutorial_srvs</exec_depend>

```

Service Server Node

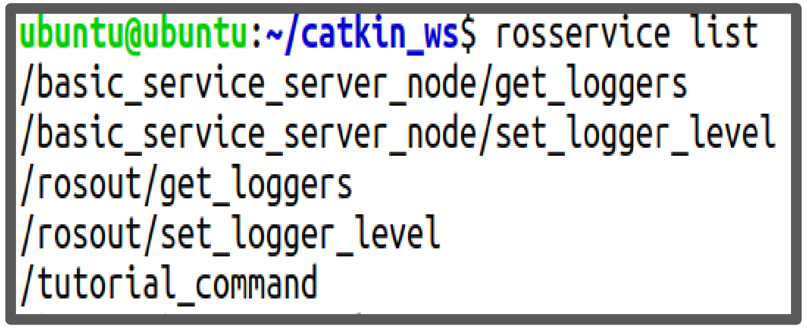
Service Server Node 생성

- Service Server Node 실행 결과



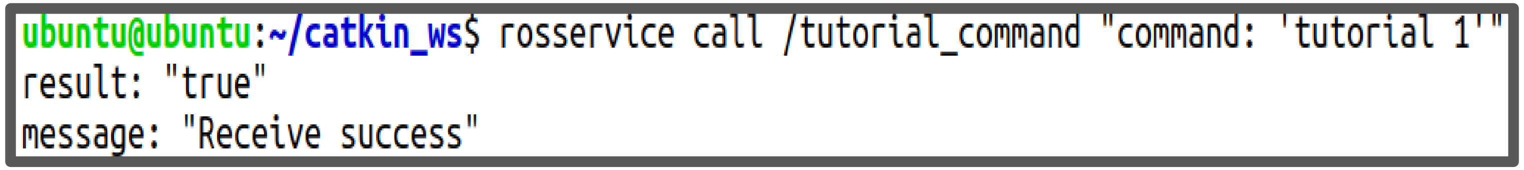
```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_service_server_tutorial basic_service_server_node
[ INFO] [1691302895.401660641]: basic_service_server_node Open
```

- Service list

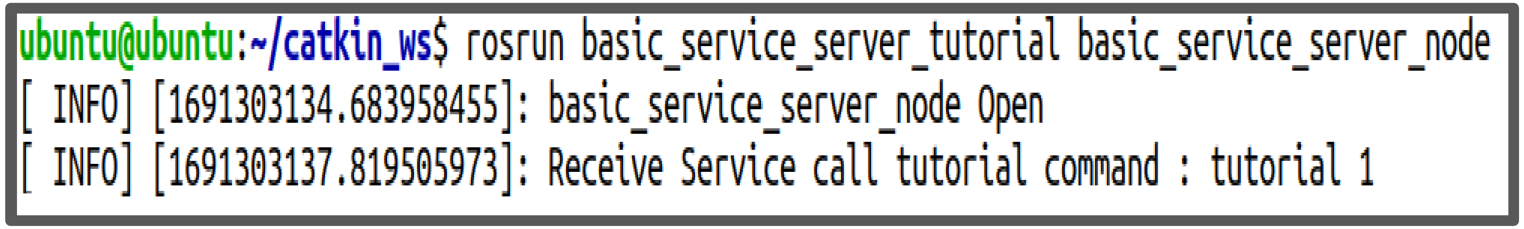


```
ubuntu@ubuntu:~/catkin_ws$ rosservice list
/basic_service_server_node/get_loggers
/basic_service_server_node/set_logger_level
/rosout/get_loggers
/rosout/set_logger_level
/tutorial_command
```

- Service 요청



```
ubuntu@ubuntu:~/catkin_ws$ rosservice call /tutorial_command "command: 'tutorial 1'"
result: "true"
message: "Receive success"
```



```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_service_server_tutorial basic_service_server_node
[ INFO] [1691303134.683958455]: basic_service_server_node Open
[ INFO] [1691303137.819505973]: Receive Service call tutorial command : tutorial 1
```

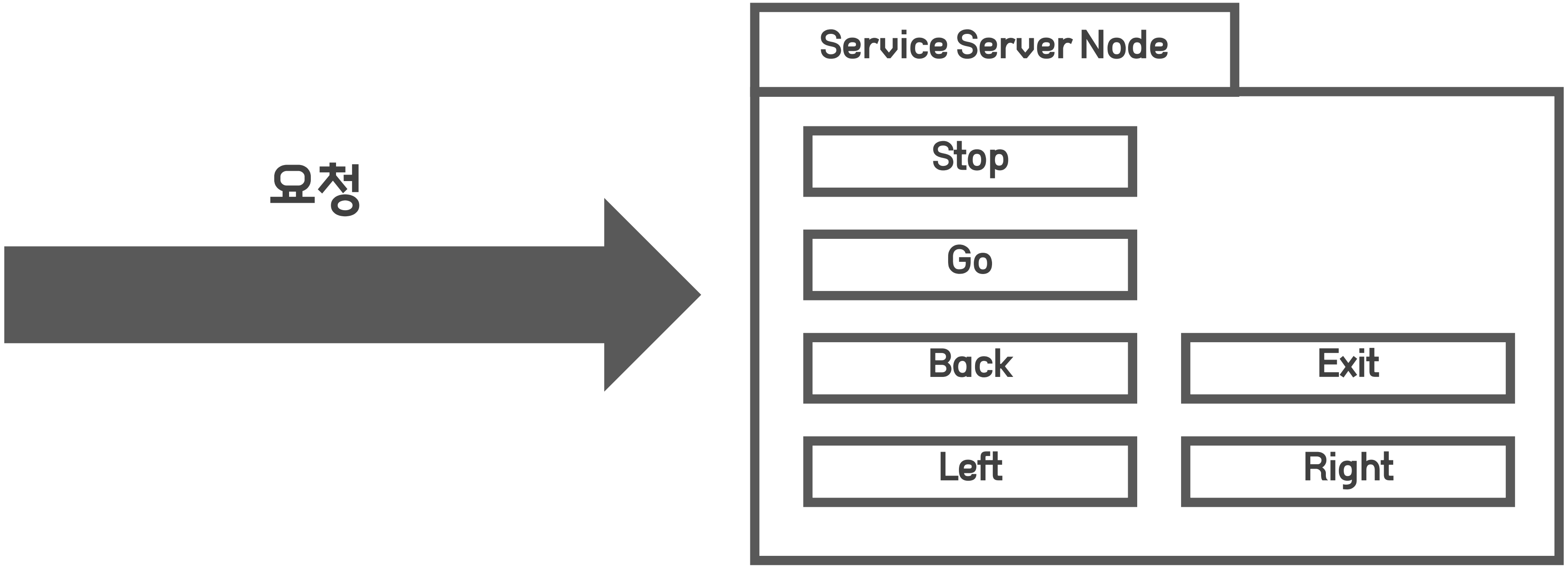
1. ROS Service 기초
2. Service 패키지 생성
3. Service Server Node
4. **Service Client Node**

Service Client Node

Service Client Node란?

서비스 통신에서 서비스 요청을 담당

동영상 강의 - Service Client Node 작성
https://youtu.be/HXufFgD0yn8?si=MfKftlByj4zCfK4P



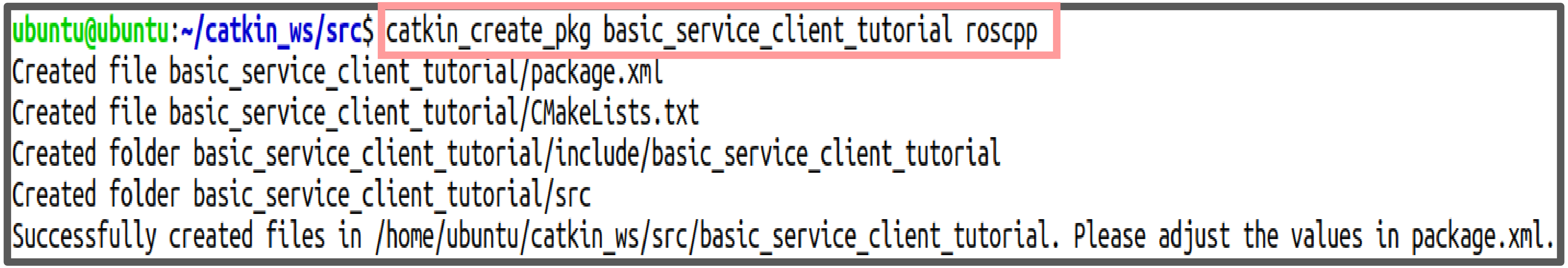
로봇의 리모콘을 누르는 역할이라고 생각하면 이해하기 쉽다.

Service Client Node

Service Client Node 생성

1) 패키지 생성

```
$ catkin_create_pkg basic_service_client_tutorial roscpp
```



```
ubuntu@ubuntu:~/catkin_ws/src$ catkin_create_pkg basic_service_client_tutorial roscpp
Created file basic_service_client_tutorial/package.xml
Created file basic_service_client_tutorial/CMakeLists.txt
Created folder basic_service_client_tutorial/include/basic_service_client_tutorial
Created folder basic_service_client_tutorial/src
Successfully created files in /home/ubuntu/catkin_ws/src/basic_service_client_tutorial. Please adjust the values in package.xml.
```

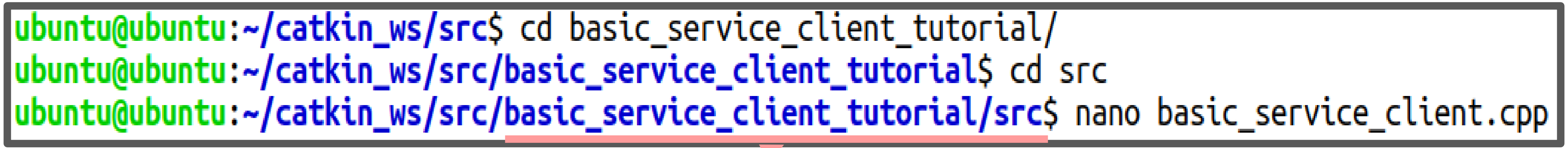
소스코드 – basic_service_client_tutorial
https://github.com/PigeonSensei/pigeon_ros_tutorial/tree/master/basic/basic_service_client_tutorial

Service Client Node

Service Client Node 생성

2) 소스코드 파일 생성

```
$ nano basic_service_server.cpp
```



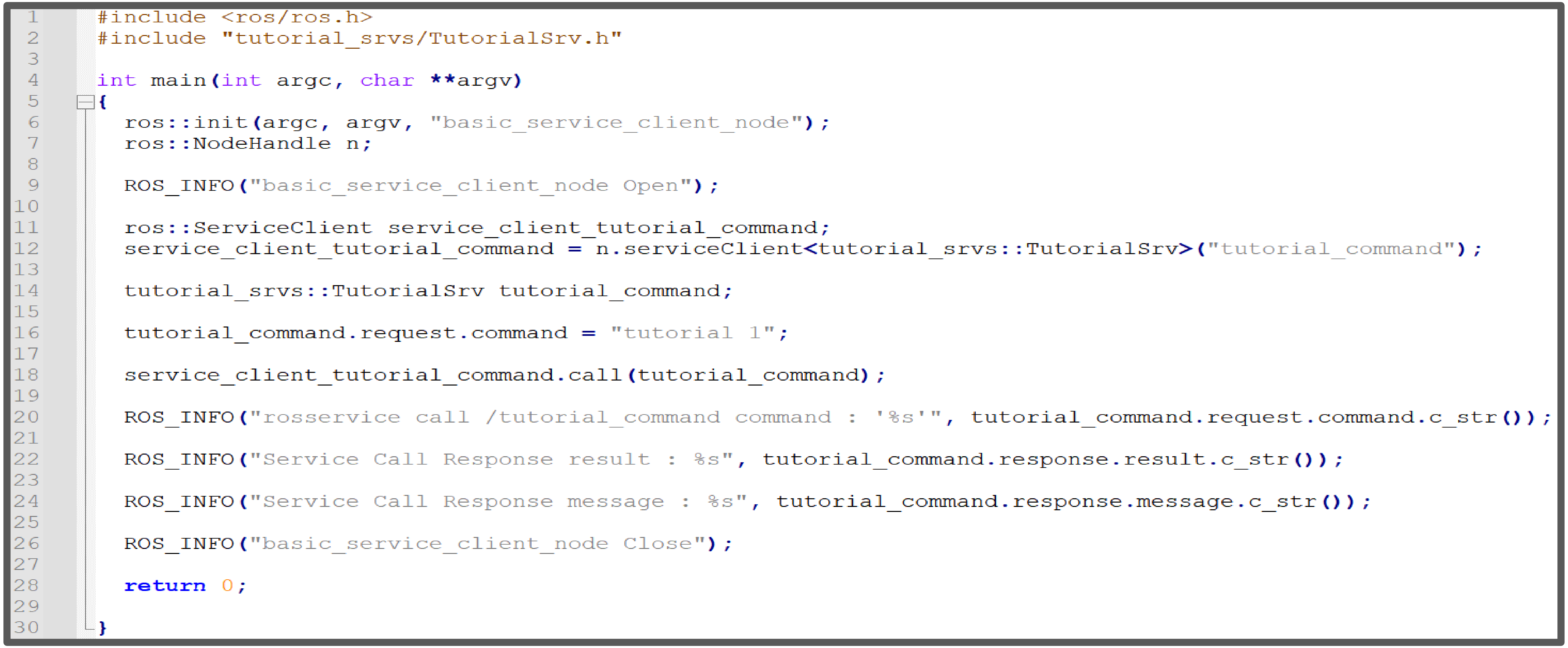
```
ubuntu@ubuntu:~/catkin_ws/src$ cd basic_service_client_tutorial/
ubuntu@ubuntu:~/catkin_ws/src/basic_service_client_tutorial$ cd src
ubuntu@ubuntu:~/catkin_ws/src/basic_service_client_tutorial/src$ nano basic_service_client.cpp
```

∵ 소스코드는 패키지의 src 경로에서 생성되어야 한다.

Service Client Node

Service Client Node 생성

3) 소스코드 작성



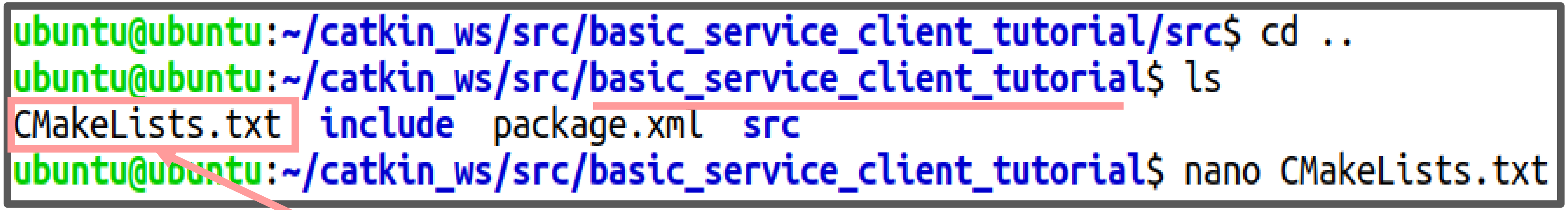
```
#include <ros/ros.h>
#include "tutorial_srvs/TutorialSrv.h"

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

    ROS_INFO("basic_service_client_node Open");

    ros::ServiceClient service_client_tutorial_command;
    service_client_tutorial_command = n.serviceClient<tutorial_srvs::TutorialSrv>("tutorial_command");

    tutorial_srvs::TutorialSrv tutorial_command;

    tutorial_command.request.command = "tutorial 1";

    service_client_tutorial_command.call(tutorial_command);

    ROS_INFO("rosservice call /tutorial_command command : '%s'", tutorial_command.request.command.c_str());

    ROS_INFO("Service Call Response result : %s", tutorial_command.response.result.c_str());

    ROS_INFO("Service Call Response message : %s", tutorial_command.response.message.c_str());

    ROS_INFO("basic_service_client_node Close");

    return 0;

}
```

Service Client Node

Service Client Node 생성

4) CMakeLists.txt 편집

```
$ nano CMakeLists.txt
```



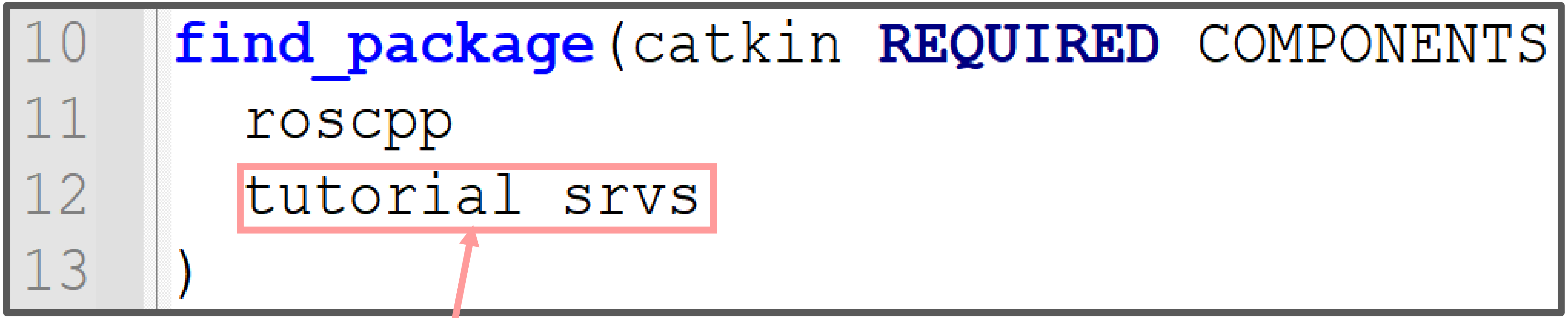
```
ubuntu@ubuntu:~/catkin_ws/src/basic_service_client_tutorial/src$ cd ..
ubuntu@ubuntu:~/catkin_ws/src/basic_service_client_tutorial$ ls
CMakeLists.txt  include  package.xml  src
ubuntu@ubuntu:~/catkin_ws/src/basic_service_client_tutorial$ nano CMakeLists.txt
```

∵ CMakeLists.txt는 패키지 경로에 있다.

Service Client Node

Service Client Node 생성

5) CMakeLists.txt의 find_package 내용 추가



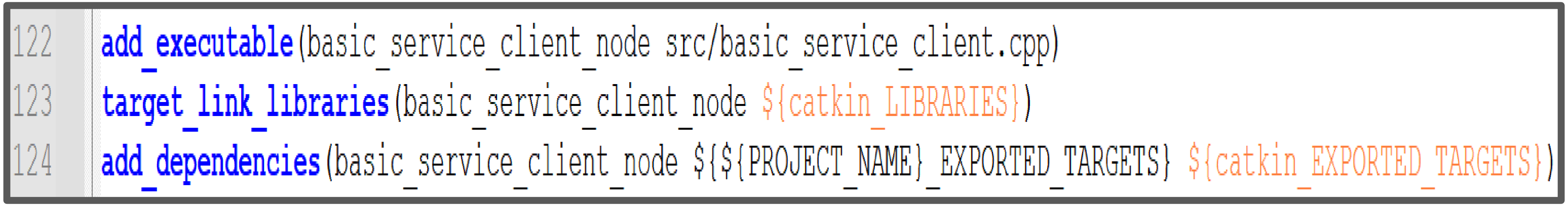
```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  tutorial srvs
)
```

∵ tutorial_srvs는 서비스 패키지 이름이다.

Service Client Node

Service Client Node 생성

6) CMakeLists.txt add_executable, target_link_librariese, add_dependencies 추가

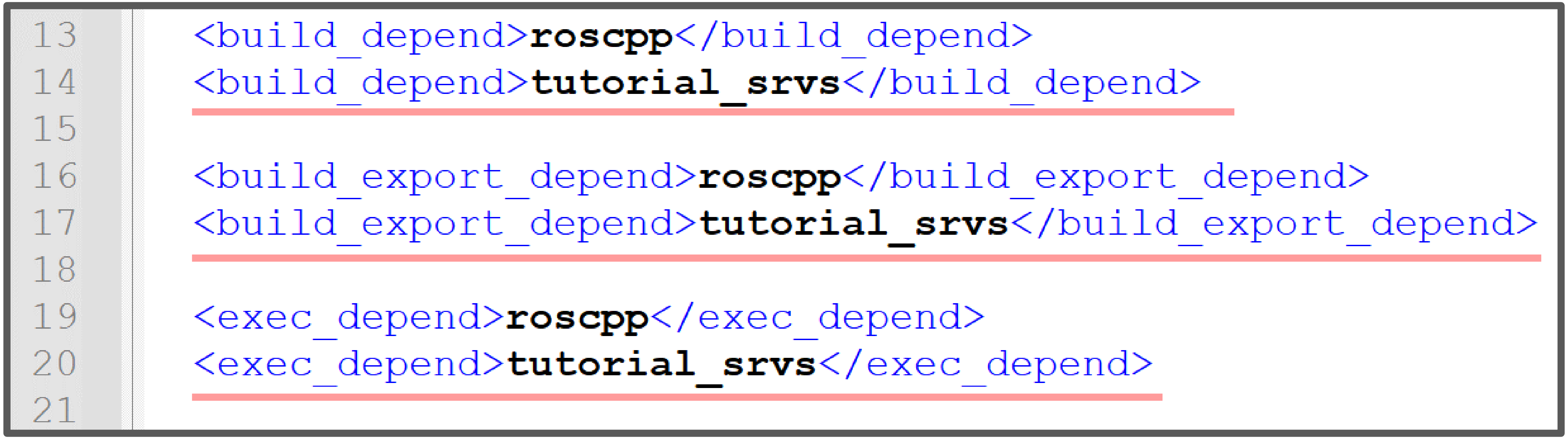


```
add_executable(basic_service_client_node src/basic_service_client.cpp)
target_link_libraries(basic_service_client_node ${catkin_LIBRARIES})
add_dependencies(basic_service_client_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
```

Service Client Node

Service Client Node 생성

7) CMakelists.txt의 find_packag에 추가 된 내용을 package.xml에 추가

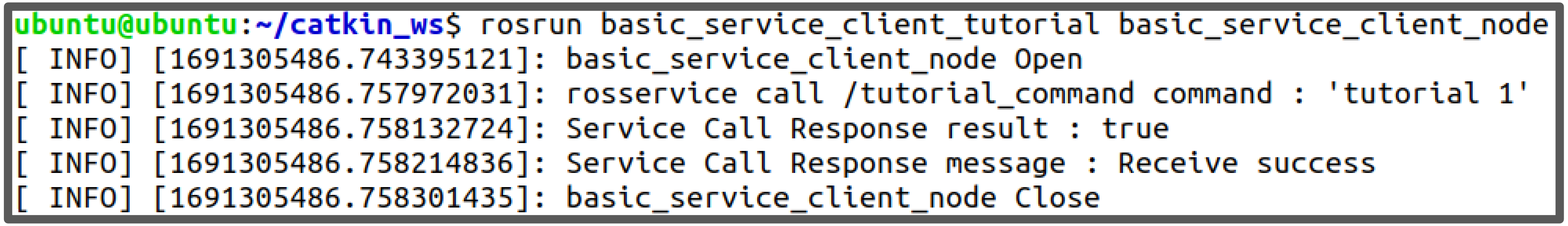


```
    <build_depend>roscpp</build_depend>
    <build_depend>tutorial_srvs</build_depend>

    <build_export_depend>roscpp</build_export_depend>
    <build_export_depend>tutorial_srvs</build_export_depend>

    <exec_depend>roscpp</exec_depend>
    <exec_depend>tutorial_srvs</exec_depend>

```

Service Client Node

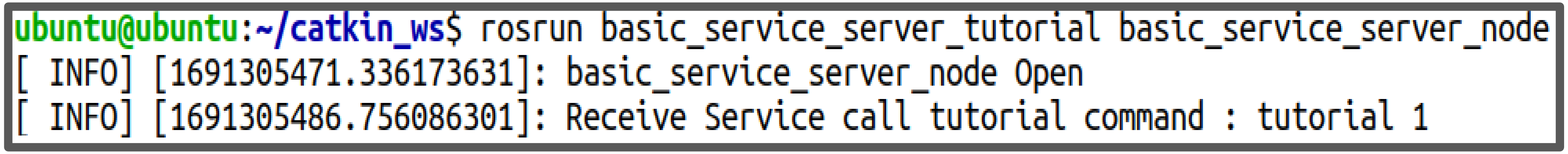
Service Client Node 생성

- Service Client Node 실행 결과



```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_service_client_tutorial basic_service_client_node
[ INFO] [1691305486.743395121]: basic_service_client_node Open
[ INFO] [1691305486.757972031]: rosservice call /tutorial_command command : 'tutorial 1'
[ INFO] [1691305486.758132724]: Service Call Response result : true
[ INFO] [1691305486.758214836]: Service Call Response message : Receive success
[ INFO] [1691305486.758301435]: basic_service_client_node Close
```

- Service Server Node에서의 반응



```
ubuntu@ubuntu:~/catkin_ws$ rosrun basic_service_server_tutorial basic_service_server_node
[ INFO] [1691305471.336173631]: basic_service_server_node Open
[ INFO] [1691305486.756086301]: Receive Service call tutorial command : tutorial 1
```

감사합니다

구선생 로보틱스