1. 实验名称及目的

1.1. 实验名称

ROS1 环境部署及安装测试

1.2. 实验目的

主要讲解如何对 ROS1 环境进行安装与配置，以及在安装后如何判断环境是否安装正常。

1.3. 关键知识点

无

2. 实验效果

能够正确判断环境是否安装正确

3. 文件目录

例程目录：[安装目录]\RflySimAPIs\2.RflySimUsage\0.ApiExps\e8_RosInstall\

文件夹/文件名称	说明
install-ROS-ubuntu.sh	环境配置文件
RosSwitch.bat	ROS 环境切换脚本
WslGUI.bat	GUI 窗口启动脚本

4. 运行环境

序号	软件要求	硬件要求	
		名称	数量
1	Windows 10 及以上版本	笔记本/台式电脑①	1
2	RflySim 工具链		

① ：推荐配置请见： https://rflysim.com/doc/zh/1/InstallLearn.html

5. 实验步骤

5.1. Ros1 环境的安装与配置

首先，我们需要找到一台空白的 Ubuntu 电脑或虚拟机，将该实验文件夹进行拷贝。之后，需要在该文件夹下打开终端，运行 install-ROS-ubuntu.sh 脚本，运行命令./ install-ROS-ubuntu.sh，运行命令后，便会进行 Ros1 环境的安装。

在安装完成之后，需要输入如下命令：

```
sudo gedit ~/.bashrc
```

注：gedit 也可以换成 vim。

打开文件后，在最后面添加上：source /opt/ros/noetic/setup.bash，之后，保存并退出。

之后，在终端中输入：

```
source ~/.bashrc
```

这样，便能够加载 ROS1 的环境，后续 ROS1 的环境就能自动加载。

注：如果不方便配置自己的 Ubuntu，也可以使用平台中提供的 WinWSL，来测试后续的步骤。

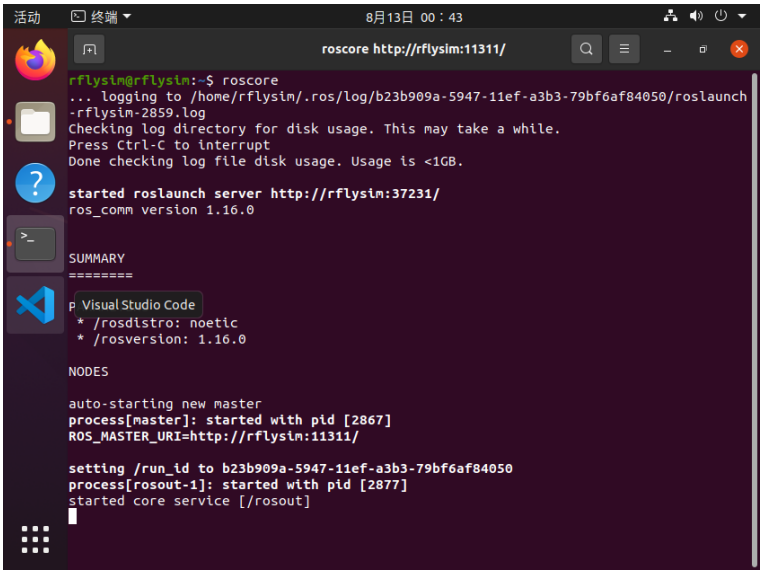
我们同样配置的有已经部署 ROS1 开发环境的虚拟机，打包放在百度网盘，如果有需要可从百度网盘中进行下载使用。链接：https://pan.baidu.com/s/15H1kFjoI-uvzn5rWAVD4iw?pwd=ijun 提取码：ijun ，配置好环境的虚拟机账户名以及密码均为：nvidia。

5.2. Ros1 环境的测试

首先，在已经安装 Ros1 的 Ubuntu 下，打开一个终端，输入：

```
roscore
```

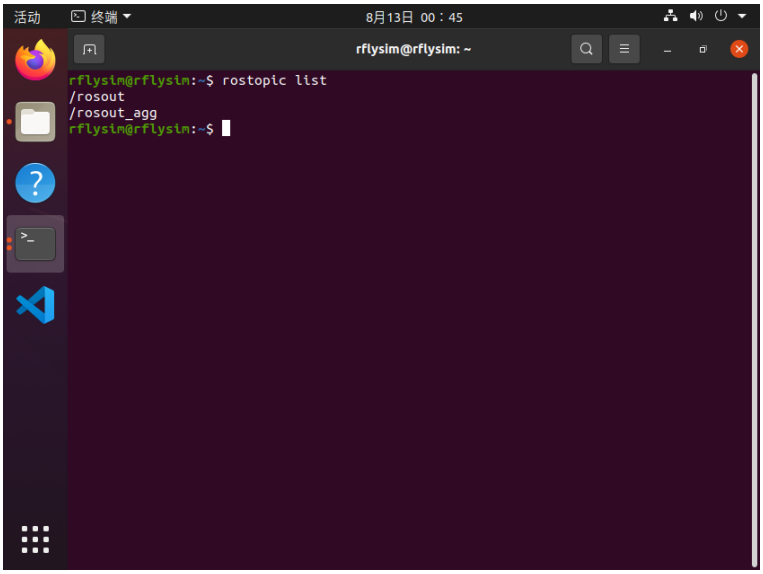
便能够启动 Ros1。



然后另起一个终端，输入：

```
rostopic list
```

就能够看到当前的 ros 消息。



如果都能正常运行，如截图显示，则说明环境没有问题。

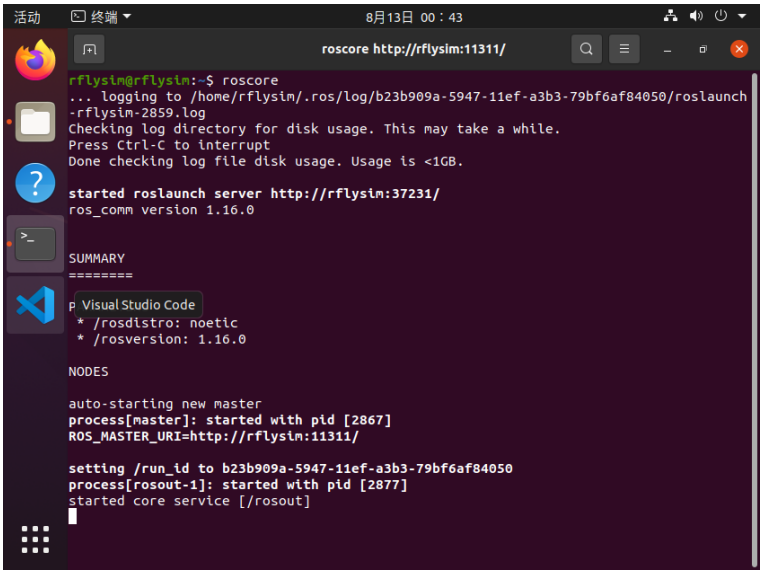
5.3. 经典海龟控制 demo

Step1：

在 Ubuntu 系统下，按下按下 CTRL+ALT+T 的快捷键，打开一个终端。输入：

```
roscore
```

来启动 ROS Master。

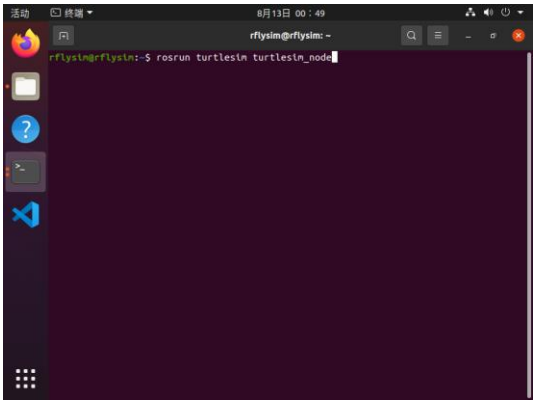


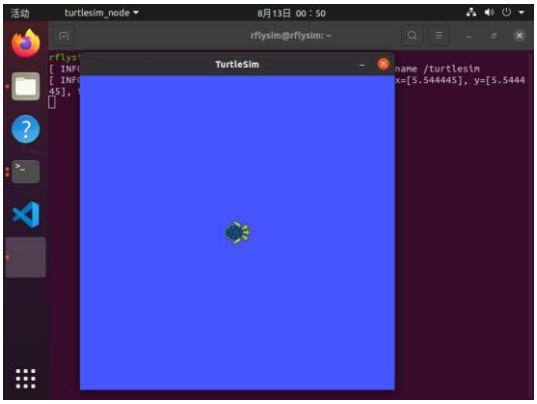
Step2：

之后，同样按下 CTRL+ALT+T 的快捷键，打开一个新终端。输入：

```
rosrun turtlesim turtlesim_node
```

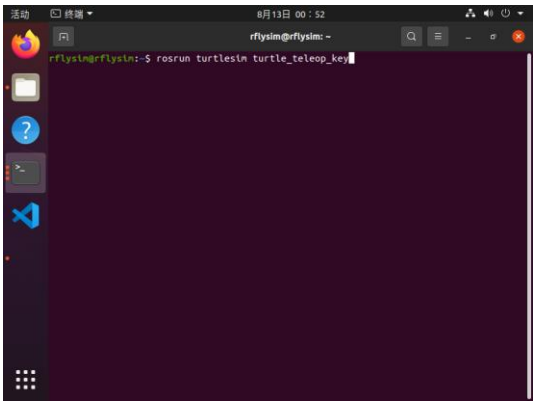
来启动小海龟仿真器。

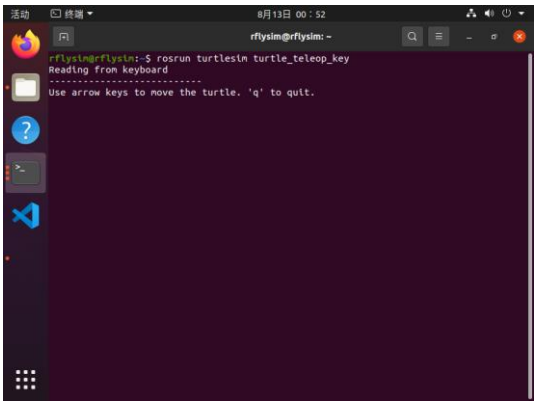




再次打开一个新的终端。输入：

```
rosrun turtlesim turtle_teleop_key
```

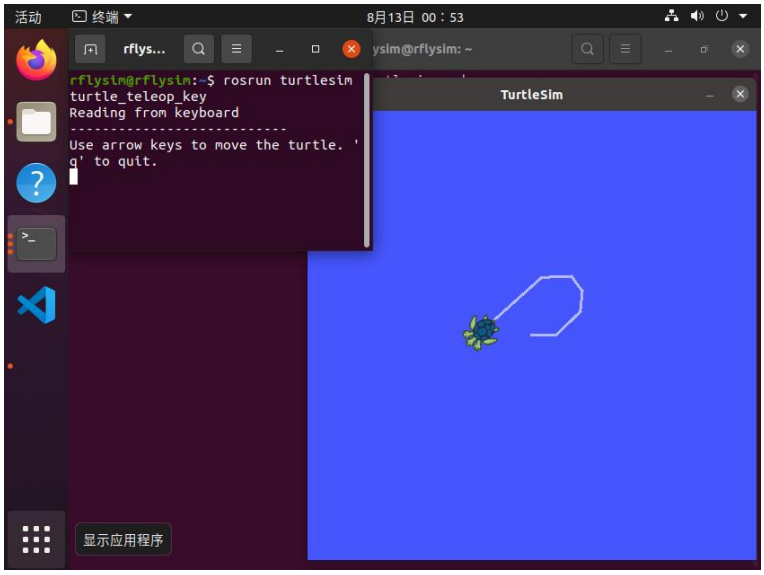




启动海龟键盘控制节点。

Step3：

在该节点中，可以通过控制键盘方向键，来控制海龟的运动。（其中上下键控制前进后退，左右键控制方向）



注：同样可以通过 WinWSL 来完成实验，步骤如下：

1.运行本文件夹中的“RosSwitch.bat”脚本，确认当前 ROS 环境，如果不是 ROS1，则切换到 ROS1。

2.运行文件夹中的“WslGUI.bat”，打开一个 GUI 窗口。

3.在 GUI 窗口中，按下 CTRL+ALT+T 的快捷键，打开一个终端。输入：

roscore

来启动 ROS Master。

4. 按下 CTRL+ALT+T 的快捷键，打开一个新终端。输入

rosrun turtlesim turtlesim_node

来启动小海龟仿真器。

5. 按下 CTRL+ALT+T 的快捷键，打开一个新终端。输入

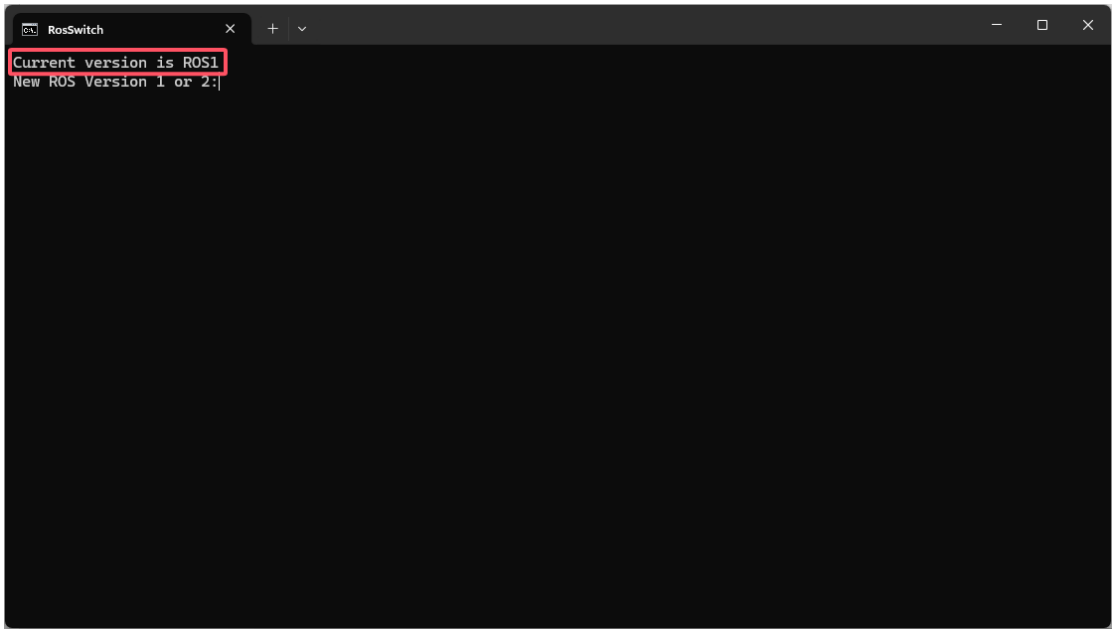
rosrun turtlesim turtle_teleop_key

启动海龟键盘控制节点。通过键盘方向键，就能控制海龟运动了。（其中上下键控制前进后退，左右键控制方向）

5.4. ROS 例程 C++版（WinWSL，必做）

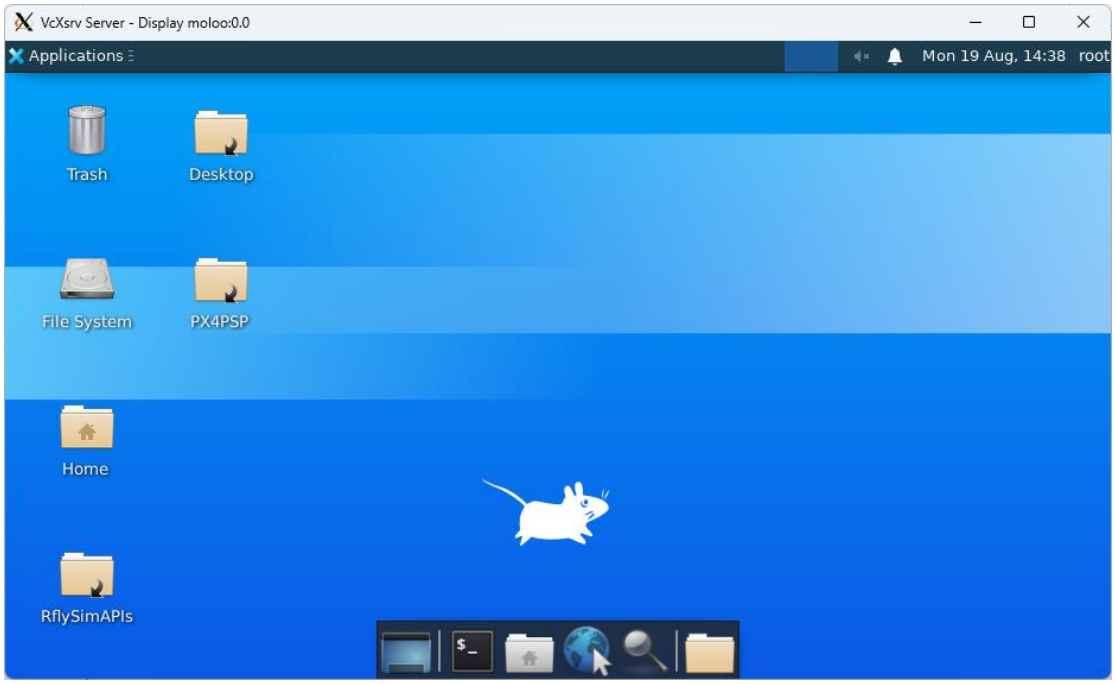
Step1：

确认环境。打开“桌面\RflyTools\RosSwitch”，查看自己当前的 ROS 环境。请确认当前环境为 ROS1，如果不是 ROS1，请输入 1 并按回车，切换为 ROS1 环境。



Step2：

打开“桌面\RflyTools\WslGUI”，为 WinWSL 的图形化界面。



Step3：

创建工作空间并初始化。打开命令行窗口输入如下的命令，创建工作空间并初始化。

```
mkdir -p 自定义空间名称/src
cd 自定义空间名称
catkin_make
```

例如，这里将自定义空间名称命名为“ros1_ws”，则上面三条命令改入如下：

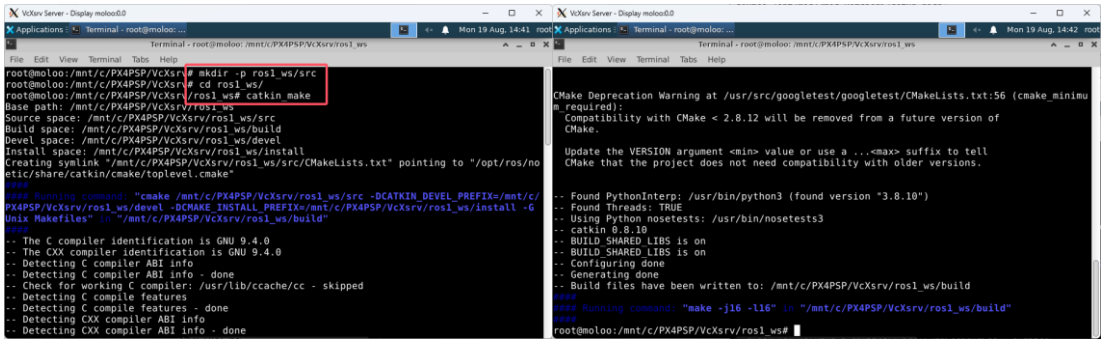
```
mkdir -p ros1_ws/src
cd ros1_ws
catkin_make
```

如果 catkin_make 出现未安装的问题，运行下面这条命令就可以解决此类问题。

```
source /opt/ros/<ros-version>/setup.bash
```

其中“<ros-version>”是安装的 ros1 的版本，例如这里使用的是 noetic 版本，则命令应该改为：

```
source /opt/ros/noetic/setup.bash
```

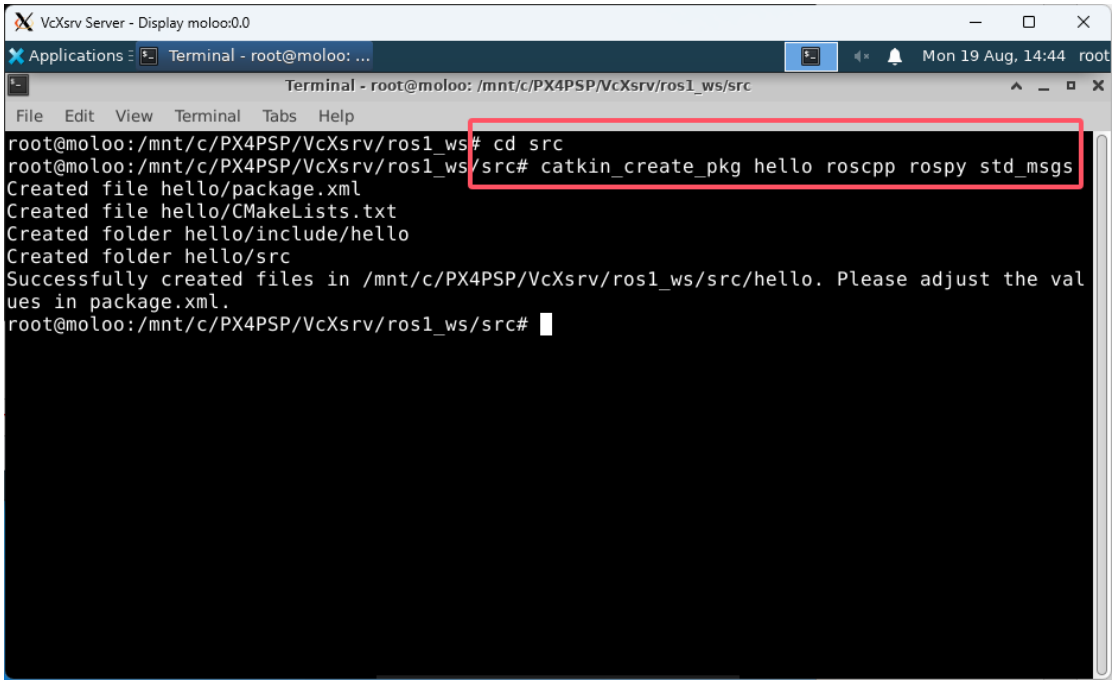


Step4：

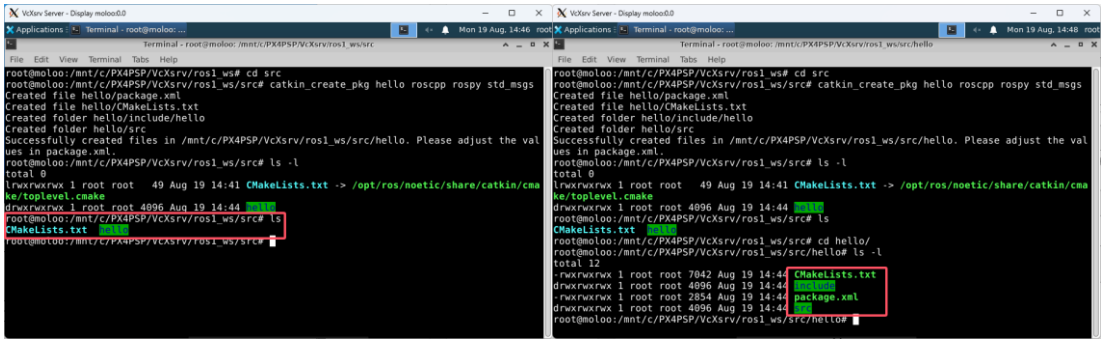
创建一个功能包。

```
cd src
catkin_create_pkg hello roscpp rospy std_msgs
```

自定义 ROS 包名 hello 一般是我们要实现的功能包的名字。



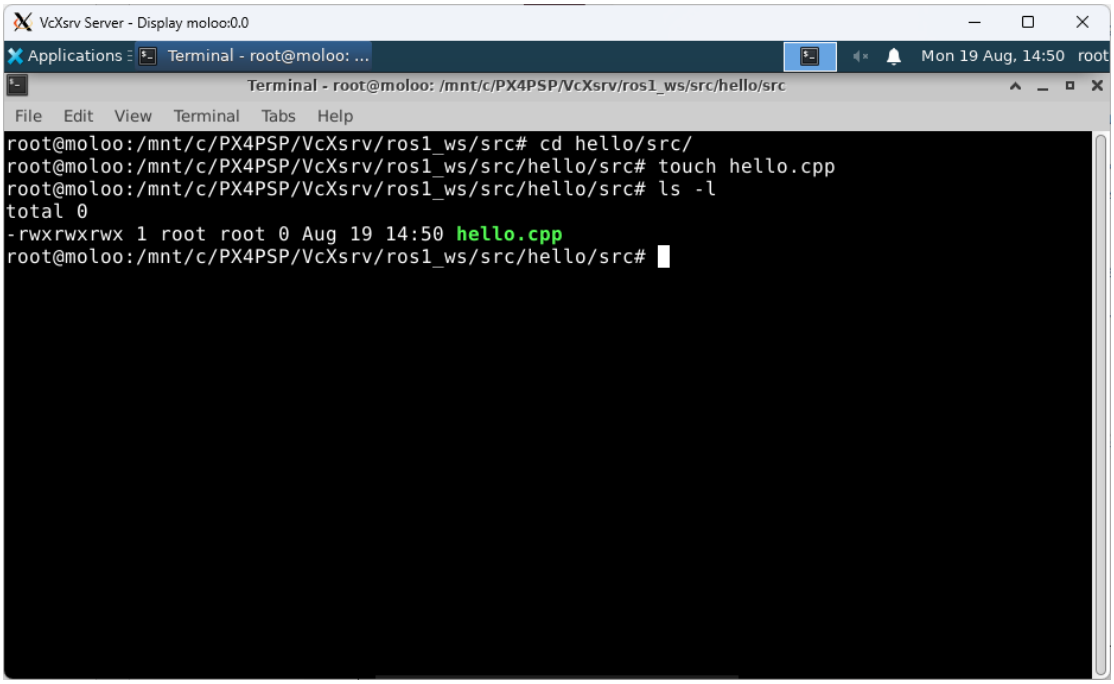
此步骤成功之后，会出现 CMakeLists 文件和 hello 包文件夹。在 hello 包文件夹下，会出现 src 和 include 文件目录。接下来在 hello/src 实现我们的功能—添加文件 hello.cpp。



Step5：编辑源文件

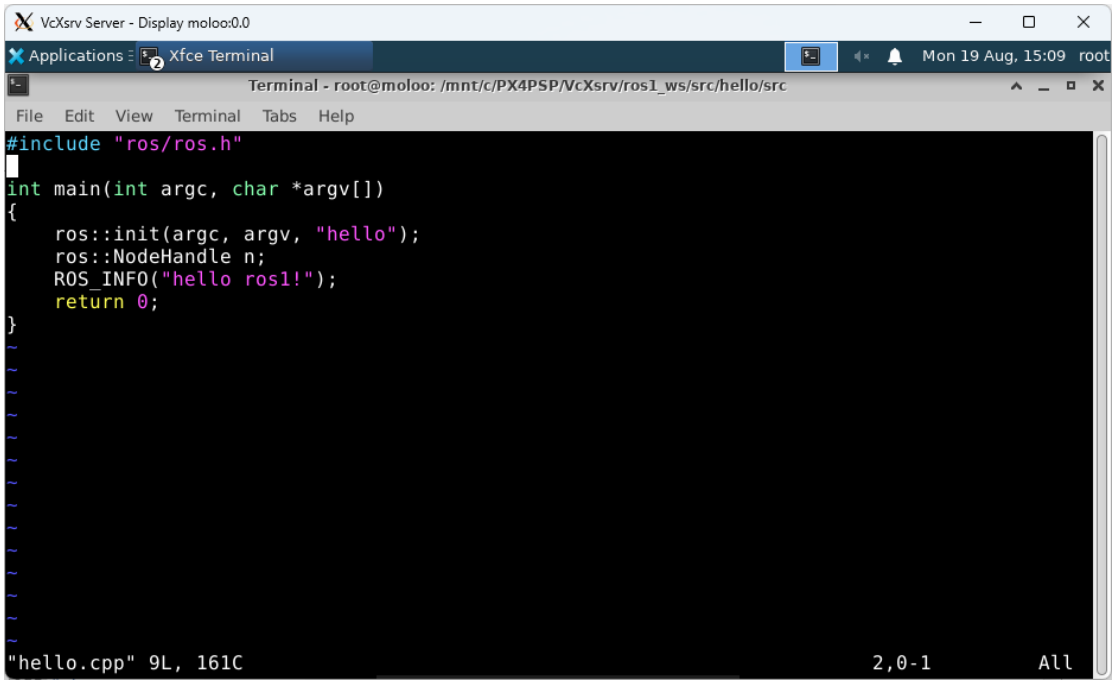
首先要创建一个文件。

```
cd hello/src
touch hello.cpp
```



然后使用 vim 或 gedit 将下面的代码写入进去，也可以直接复制例程目录中的代码。

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

int main(int argc, char *argv[])
{
    //执行 ros 节点初始化
    ros::init(argc,argv,"hello");
    //创建 ros 节点句柄(非必须)
    ros::NodeHandle n;

    ROS_INFO("hello!");

    return 0;
}
```



Step6：编辑配置文件

在功能包 hello 的目录下的 CmakeLists.txt 文件中修改配置信息。如果下面的 gedit 命令报错，说明没有安装 gedit，运行“apt install gedit”进行安装。

```
cd ..
gedit CmakeLists.txt
```

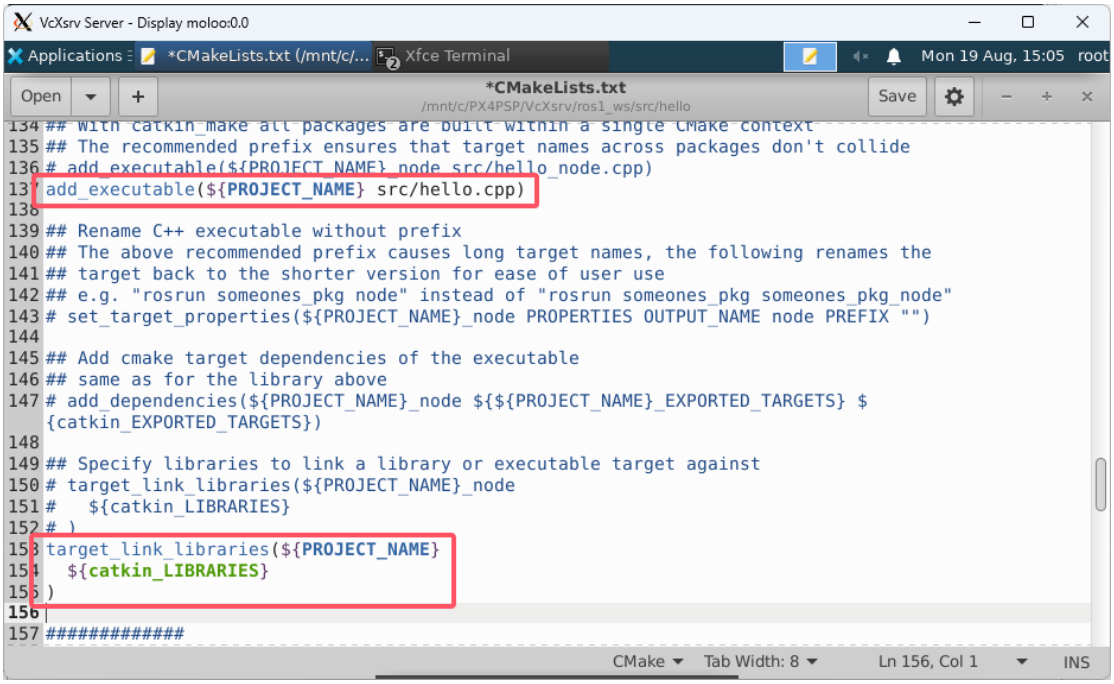
下面的代码都可以从注释中复制得到。下面这部分代码是告诉 CMake 哪些源文件应该编译并链接在一起生成一个可执行文件。

```
add_executable(要生成的目标文件
  src/源文件.cpp
)
```

下面这部分代码是 CMake 中的一个指令，用于指定一个目标（例如一个可执行文件或库）需要链接的外部库或内部库。这里的 ${catkin_LIBRARIES} 包含了编译时需要的所有 ROS 相关的库和依赖项。

```
target_link_libraries(要生成的目标文件
  ${catkin_LIBRARIES}
)
```

下面是修改后的截图，注意注释中的代码在命名后面都追加了“_node”，为方便实验以及符合前面的命名，这里把“_node”全部进行了删除。否则，在运行的时候节点名称后面要加“_node”。

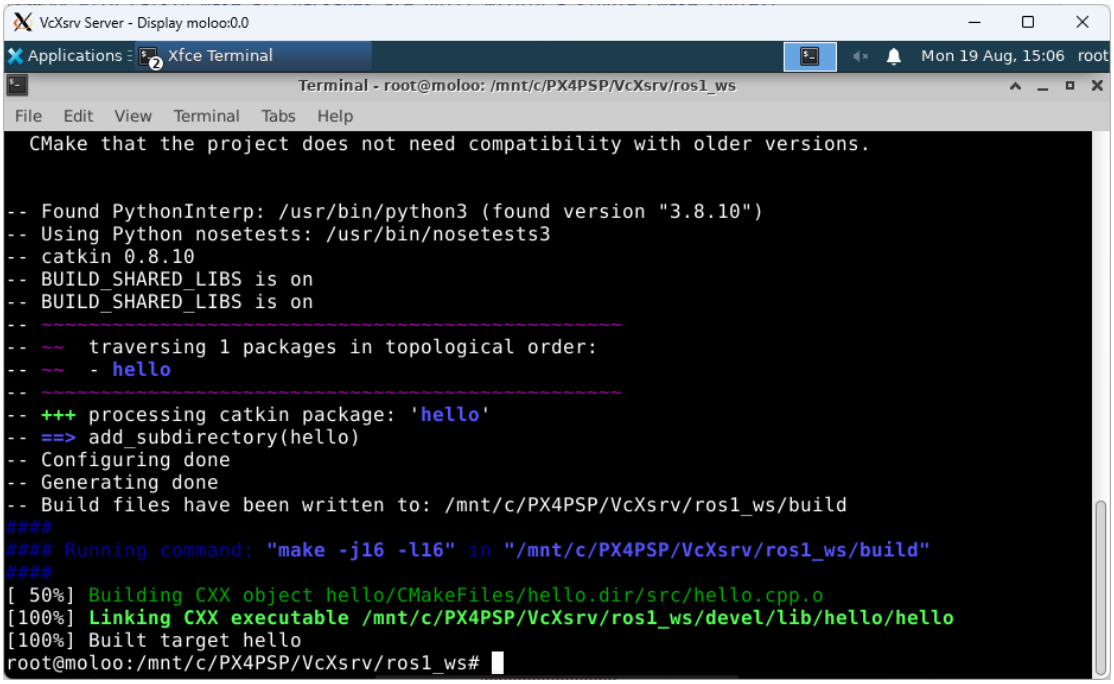


修改完成之后 在工作空间下面执行 catkin_make 进行编译。

Step7：编译

修改完成之后 在工作空间下面执行 catkin_make 进行编译。下面使用了两次返回上一级路径的命令，是为了回到“ros1_ws”目录下。

```
cd ..
cd ..
catkin_make
```



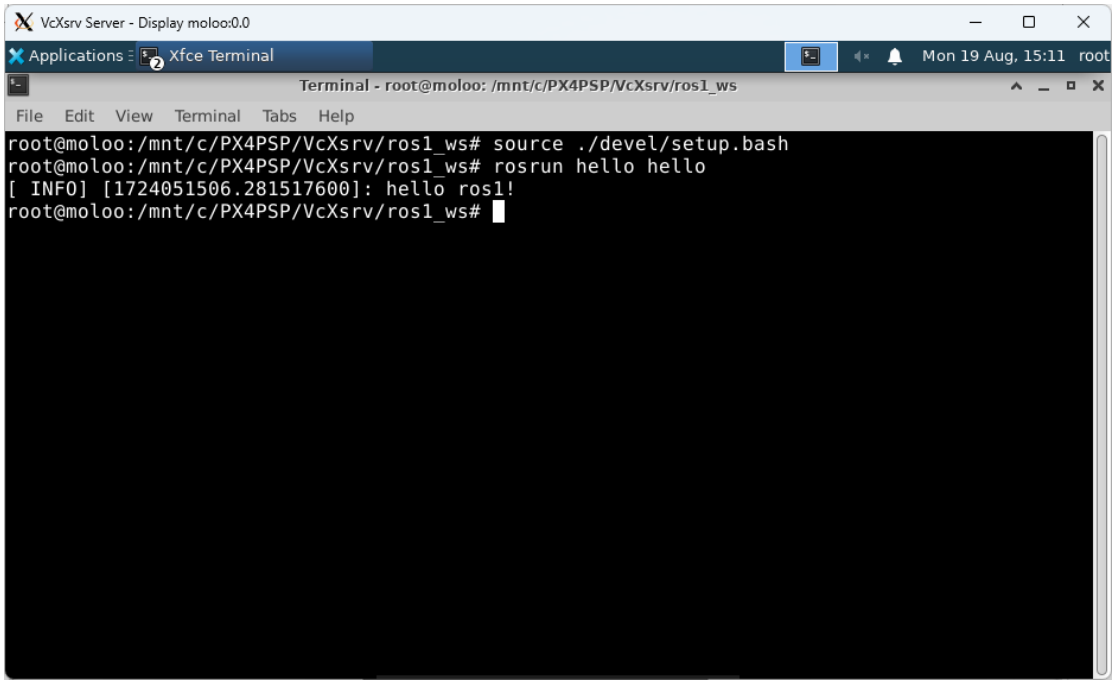
Step8：执行

下面的命令是通用的运行命令。

```
roscore
// 打卡一个新的命令窗口
cd 工作空间
source ./devel/setup.bash
rosrun 包名 目标生成的文件
```

根据自定义的命名，上面的代码应该修改为下面的代码。

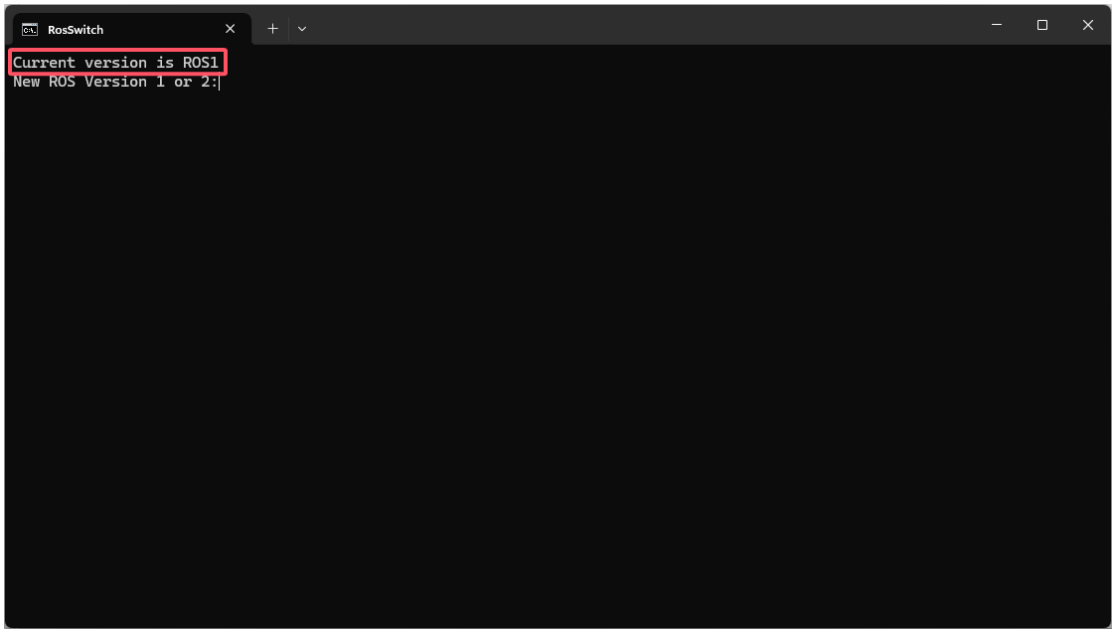
```
roscore
// 打卡一个新的命令窗口
cd ros1_ws
source ./devel/setup.bash
rosrun hello hello
```



5.5. ROS 例程 python 版（WinWSL，必做）

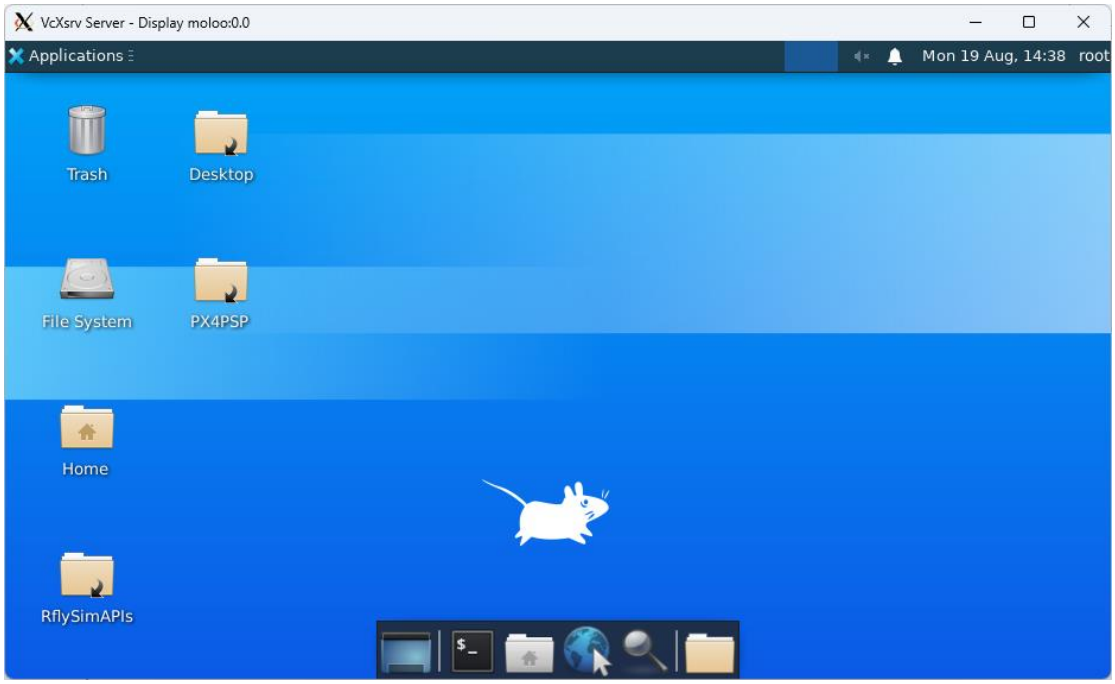
Step1：

确认环境。打开“桌面\RflyTools\RosSwitch”，查看自己当前的 ROS 环境。请确认当前环境为 ROS1，如果不是 ROS1，请输入 1 并按回车，切换为 ROS1 环境。



Step2：

打开“桌面\RflyTools\WslGUI”，为 WinWSL 的图形化界面。



Step3：创建工作空间

首先，你需要一个 ROS 工作空间，在上一节中如果创建了空间则无需继续操作后面的内容。如果你还没有，可以通过以下命令创建一个：

```
source /opt/ros/<ros-version>/setup.bash  # <ros-version> 替换为你的 ROS 版本，如 melodic 或 noetic
mkdir -p ~/ros1_ws/src
cd ~/ros1_ws/
catkin_make  # 或者使用 catkin build，如果你安装了 catkin_tools
```

根据平台的版本，上面的代码修改为如下：

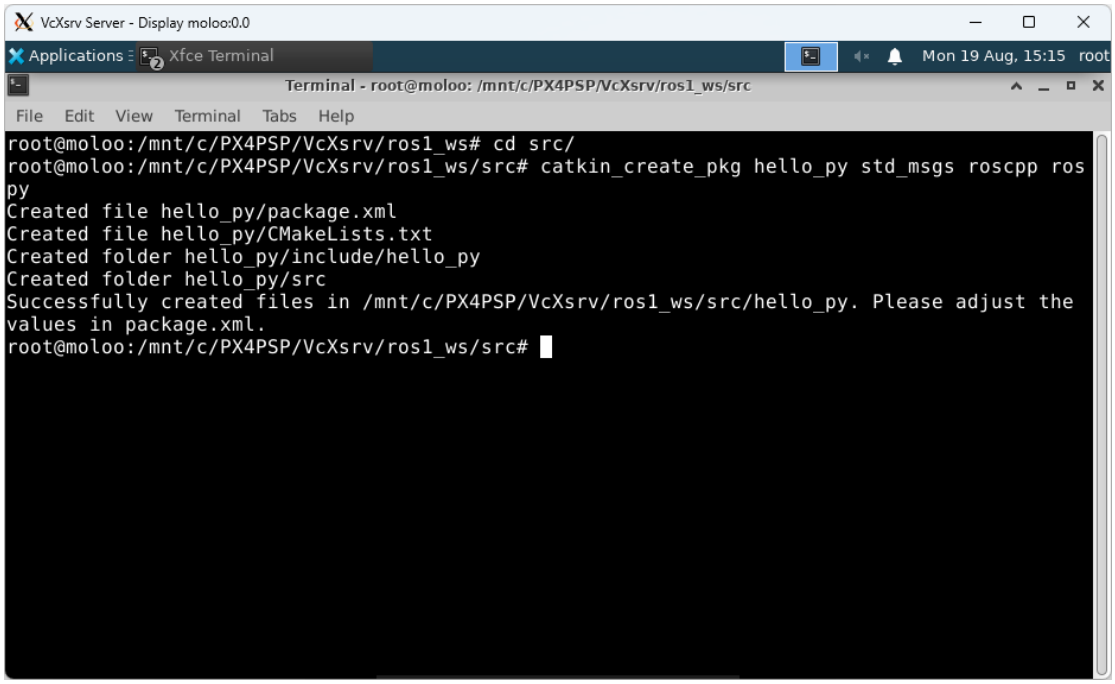
```
source /opt/ros/noetic/setup.bash
mkdir -p ros1_ws/src
```

```
cd ros1_ws
catkin_make
```

Step4：创建 ROS 包

在你的工作空间内创建一个新的 ROS 包：

```
cd src
catkin_create_pkg hello_py std_msgs roscpp rospy
```



Step5：编写 Python 代码

发布者（Publisher）

在 hello_py/scripts/目录下创建一个名为 talker.py 的文件（如果 scripts 目录不存在，请创建它），并添加以下代码：

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

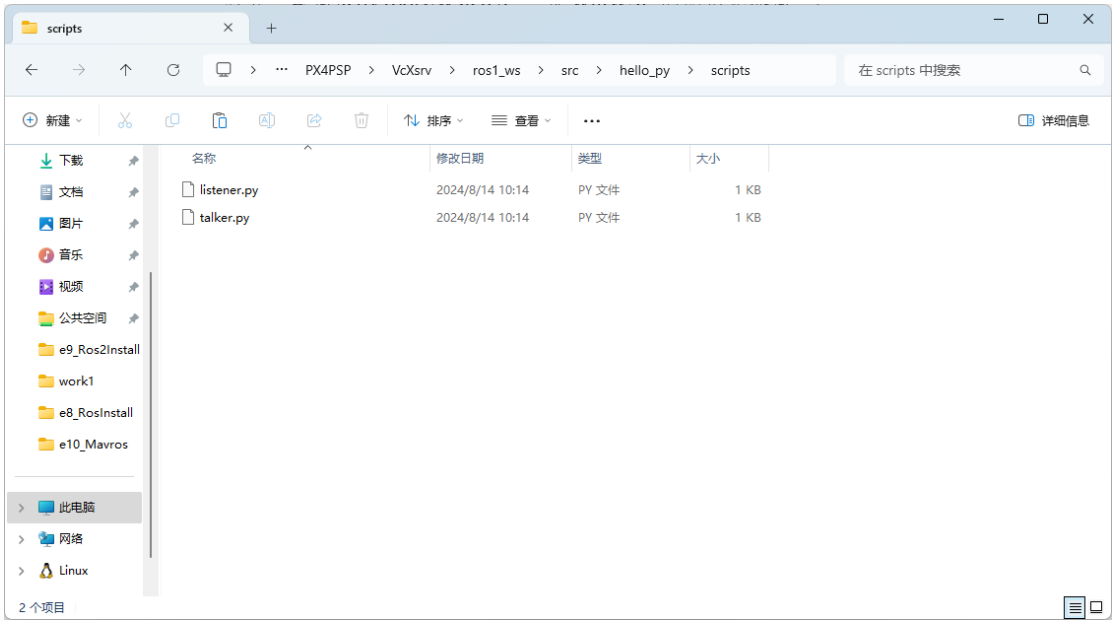
订阅者（Subscriber）

在同一个 scripts 目录下，创建一个名为 listener.py 的文件，并添加以下代码：

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String
```

```
def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

可以使用复制粘贴的方式快速的放入对应的文件夹中，该 WinWSL 中的目录对应如下图片中的 Windows 目录：



Step6：给予脚本执行权限

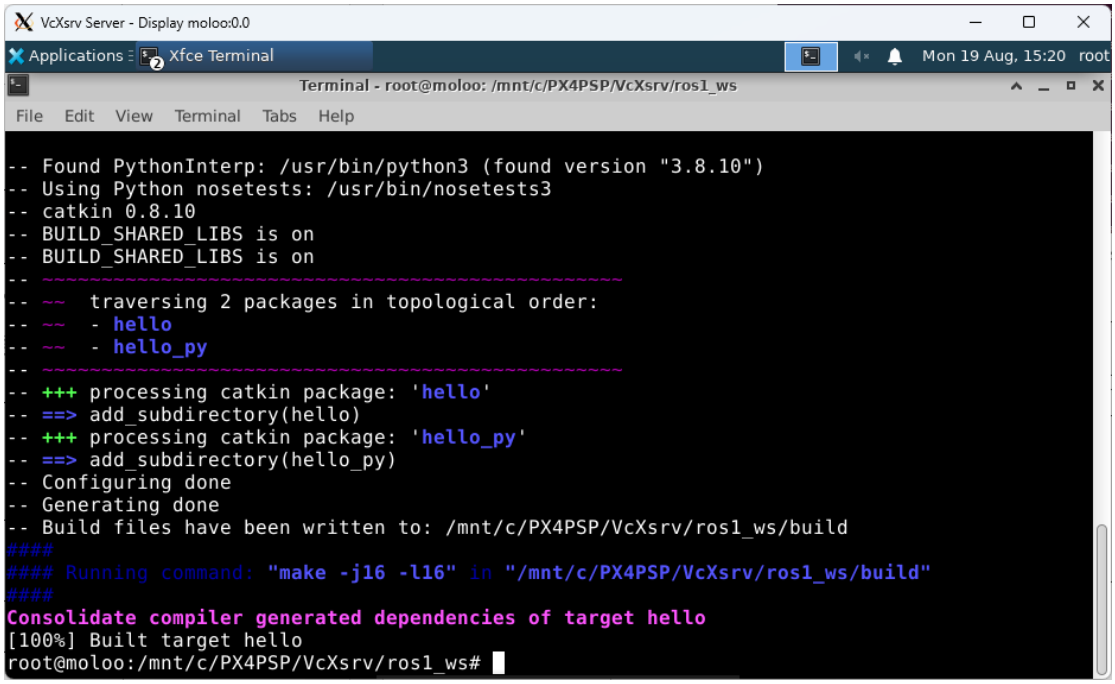
给这两个 Python 脚本执行权限：

```
chmod +x talker.py
chmod +x listener.py
```

Step7：编译 ROS 包（对于 Python 脚本其实不需要，但确保环境是最新的）

返回工作空间的根目录并编译（尽管对于 Python 脚本这步不是必需的，但确保环境是最新的）：

```
cd ..
cd ..
cd ..
catkin_make
```



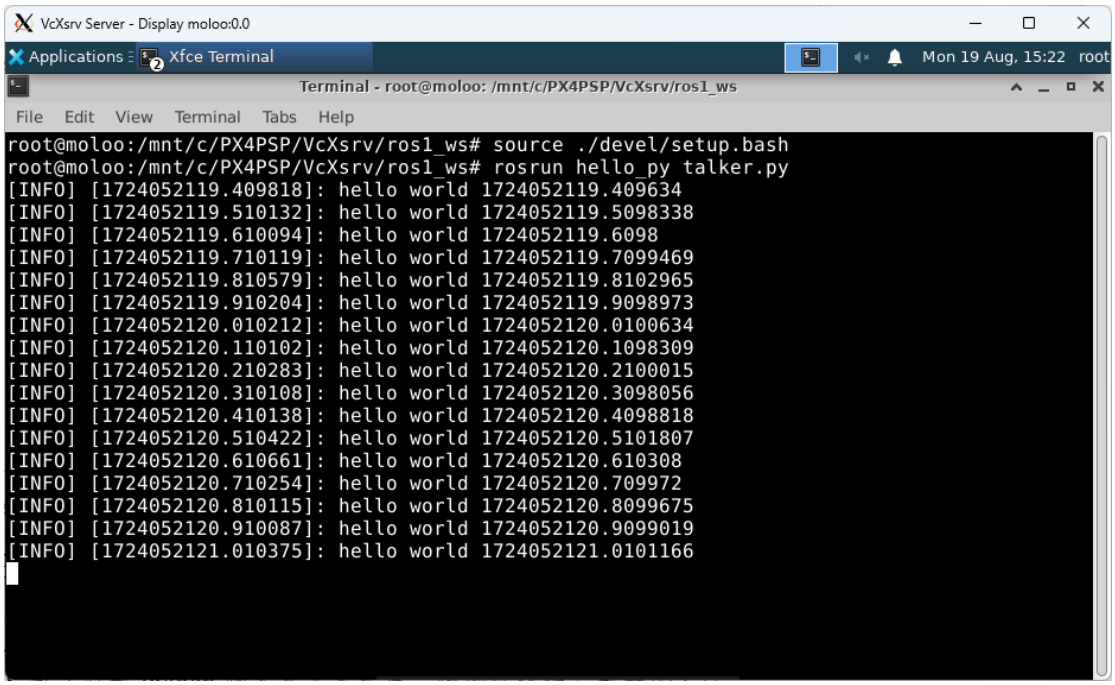
Step8：运行例程

首先，确保 ROS 核心正在运行：

```
roscore
```

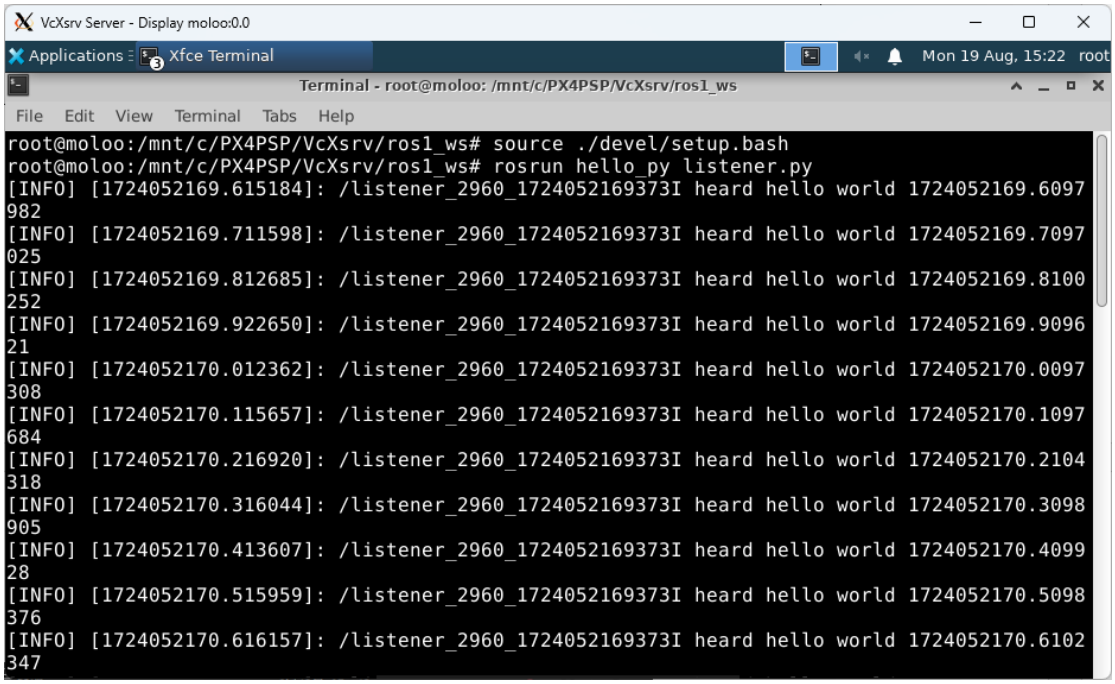
然后，在新的终端窗口中启动发布者：

```
source ./devel/setup.bash
rosrun hello_py talker.py
```



在另一个新的终端窗口中启动订阅者：

```
source ./devel/setup.bash
rosrun hello_py listener.py
```



现在你应该会在终端窗口中看到发布者发布的消息和订阅者接收到的消息。如果运行完提示：

```
/usr/bin/env: “python”: 没有那个文件或目录
```

如果是安装的平台提供的虚拟机，所有运行环境应该是，则需要设置一下软连接。

```
# 找 Python3.8 位置
whereis python3.8

# 配置软连接
cd /usr/bin
ln -s /usr/bin/python3.8 python
```

5.6. ROS 例程 C++版（虚拟机，选做）

ROS 中涉及的编程语言以 C++和 Python 为主，实现流程大致如下：

1.先创建一个工作空间；2.再创建一个功能包；3.编辑源文件；4.编辑配置文件；5.编译并执行。

Step1：创建工作空间并初始化

```
mkdir -p 自定义空间名称/src
cd 自定义空间名称
catkin_make
```

例如，这里将自定义空间名称命名为“ros1_ws”，则上面三条命令改入如下：

```
mkdir -p ros1_ws/src
cd ros1_ws
catkin_make
```

如果 catkin_make 出现未安装的问题，运行下面这条命令就可以解决此类问题。

```
source /opt/ros/<ros-version>/setup.bash
```

其中“<ros-version>”是安装的 ros1 的版本，例如这里使用的是 noetic 版本，则命令应该改为：

```
source /opt/ros/noetic/setup.bash
```

Step2：创建一个功能包

```
cd src
catkin_create_pkg hello roscpp rospy std_msgs
```

自定义 ROS 包名 hello 一般是我们要实现的功能包的名字

此步骤成功之后，会出现 src 和 include 文件目录。接下来在 hello/src 实现我们的功能—添加文件 hello.cpp。

Step3：编辑源文件

首先要创建一个文件。

```
cd hello/src
touch hello.cpp
```

然后使用 gedit 将下面的代码写入进去。

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

int main(int argc, char *argv[])
{
    //执行 ros 节点初始化
    ros::init(argc,argv,"hello");
    //创建 ros 节点句柄(非必须)
    ros::NodeHandle n;

    ROS_INFO("hello ros1!");

    return 0;
}
```

Step4：编辑配置文件

在功能包 hello 的目录下的 CmakeLists.txt 文件中修改配置信息。

```
cd ..
gedit CmakeLists.txt
```

下面的代码都可以从注释中复制得到。下面这部分代码是告诉 CMake 哪些源文件应该编译并链接在一起生成一个可执行文件。

```
add_executable(要生成的目标文件
  src/源文件.cpp
)
```

下面这部分代码是 CMake 中的一个指令，用于指定一个目标（例如一个可执行文件或库）需要链接的外部库或内部库。这里的 ${catkin_LIBRARIES} 包含了编译时需要的所有 ROS 相关的库和依赖项。

```
target_link_libraries(要生成的目标文件
  ${catkin_LIBRARIES}
)
```

下面是修改后的截图，注意注释中的代码在命名后面都追加了“_node”，为方便实验以及符合前面的命名，这里把“_node”全部进行了删除。

修改完成之后 在工作空间下面执行 catkin_make 进行编译

Step6：编译

修改完成之后 在工作空间下面执行 catkin_make 进行编译。下面使用了两次返回上一级路径的命令，是为了回到“ros1_ws”目录下。

```
cd ..
cd ..
catkin_make
```

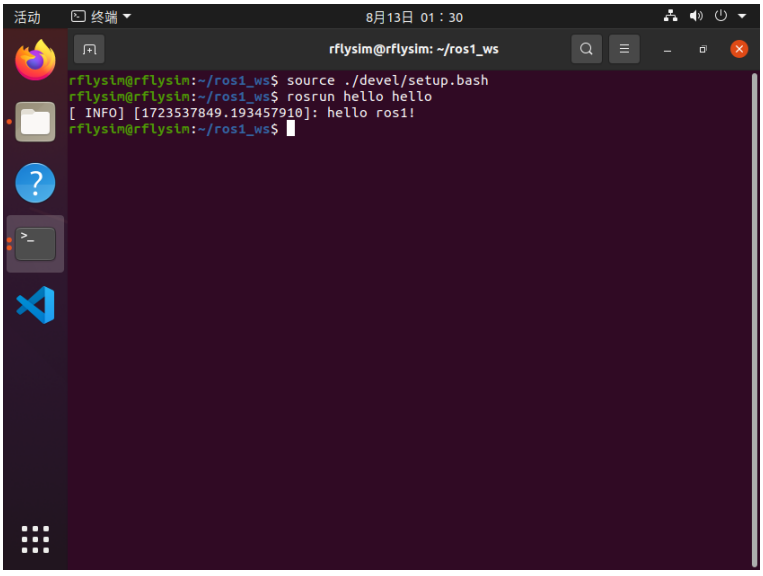
Step7：执行

下面的命令是通用的运行命令。

```
roscore
cd 工作空间
source ./devel/setup.bash
rosrun 包名 目标生成的文件
```

根据自定义的命名，上面的代码应该修改为下面的代码。

```
roscore
cd ros1_ws
source ./devel/setup.bash
rosrun hello hello
```



5.7. ROS 例程 python 版（虚拟机，选做）

Step1：创建工作空间

首先，你需要一个 ROS 工作空间，在上一节中如果创建了空间则无需继续操作后面的内容。如果你还没有，可以通过以下命令创建一个：

```
source /opt/ros/<ros-version>/setup.bash  # <ros-version> 替换为你的 ROS 版本，如 melodic 或 noetic
mkdir -p ~/ros1_ws/src
cd ~/ros1_ws/
catkin_make  # 或者使用 catkin build，如果你安装了 catkin_tools
```

根据平台的版本，上面的代码修改为如下：

```
source /opt/ros/noetic/setup.bash
mkdir -p ros1_ws/src
cd ros1_ws
catkin_make
```

Step2：创建 ROS 包

在你的工作空间内创建一个新的 ROS 包：

```
cd src
catkin_create_pkg hello_py std_msgs roscpp rospy
```

Step3：编写 Python 代码

发布者（Publisher）

在 hello_py/scripts/目录下创建一个名为 talker.py 的文件（如果 scripts 目录不存在，请创建它），并添加以下代码：

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

订阅者（Subscriber）

在同一个 scripts 目录下，创建一个名为 listener.py 的文件，并添加以下代码：

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
```

```
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

Step4：给予脚本执行权限

给这两个 Python 脚本执行权限：

```
chmod +x talker.py
chmod +x listener.py
```

Step5：编译 ROS 包（对于 Python 脚本其实不需要，但确保环境是最新的）

返回工作空间的根目录并编译（尽管对于 Python 脚本这步不是必需的，但确保环境是最新的）：

```
cd ..
cd ..
cd ..
catkin_make
```

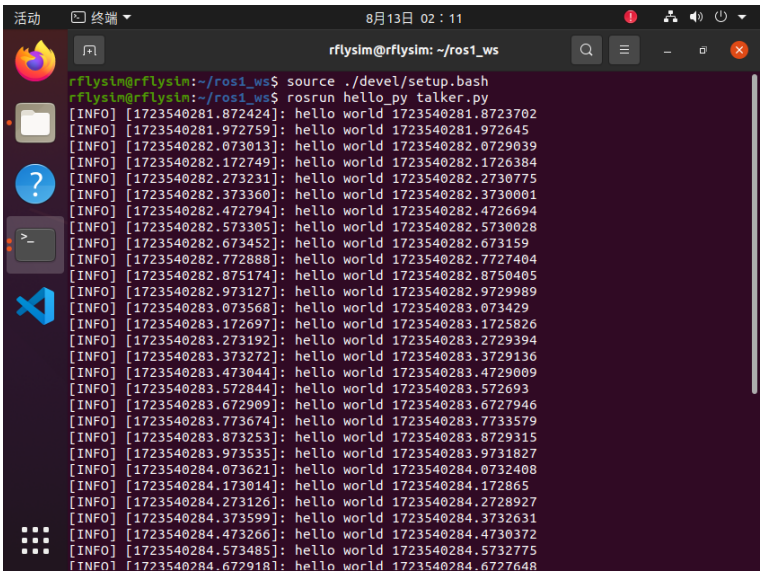
Step5：运行例程

首先，确保 ROS 核心正在运行：

```
roscore
```

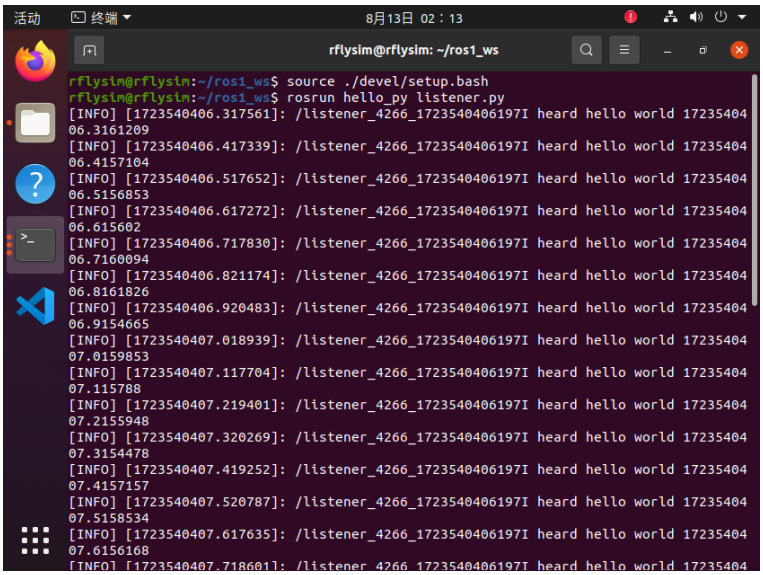
然后，在新的终端窗口中启动发布者：

```
source ./devel/setup.bash
rosrun hello_py talker.py
```



在另一个新的终端窗口中启动订阅者：

```
source ./devel/setup.bash
rosrun hello_py listener.py
```



现在你应该会在终端窗口中看到发布者发布的消息和订阅者接收到的消息。如果运行完提示：

```
/usr/bin/env: “python”: 没有那个文件或目录
```

如果是安装的平台提供的虚拟机，所有运行环境应该是，则需要设置一下软连接。

```
# 找 Python3.8 位置
whereis python3.8

# 配置软连接
cd /usr/bin
ln -s /usr/bin/python3.8 python
```

5.8. ROS 学习

ROS1（Robot Operating System 1）的学习网站众多，以下是一些推荐的学习资源，包括官方网站、教程、社区和博客等，它们为初学者和进阶用户提供了丰富的学习内容：

ROS 官方 Wiki：

地址：https://wiki.ros.org/cn/ROS/

描述：ROS 的官方 Wiki 是学习和访问 ROS 进程的主要网站，它包含了大量的教程文档和资源链接。

创客制造：

地址：https://www.ncnynl.com/

描述：这是一个针对初学者的官网，提供了 ROS+Ubuntu 集成版，方便初学者直接使用。

大学生 MOOC：

地址：https://www.icourse163.org/

描述：这是一个在线教育平台，用户可以在上面找到 ROS 相关的视频课程进行学习。

ROS 问答社区：

地址：https://answers.ros.org/questions/

描述：ROS 的问答社区，用户可以在这里提问和回答关于 ROS 的问题。

博客专栏：

地址：如 https://blog.csdn.net/zhangrelay/category_9267010.html

描述：一些专业的博客专栏，如张瑞雷 ZhangRelay 老师的博客，提供了关于 ROS 的深入教程和案例分析。

官方文档和示例：

描述：ROS 的官方文档和示例代码也是学习 ROS 的重要资源，它们提供了详细的 API 说明和使用示例。

在学习 ROS 时，建议从官方 Wiki 开始，了解 ROS 的基本概念和架构，然后通过上述网站和社区进一步深入学习。同时，也可以参考一些具体的项目案例，通过实践来加深理解和掌握。

6. 参考资料

[1]. 无。

7. 常见问题

Q:在进行 ROS 环境部署失败？

N:检查网络是否正常，在可能的情况下，尝试使用代理再次运行。