1.实验名称及目的

1.1 实验名称

ROS2环境部署及安装测试

1.2 实验目的

主要讲解如何对ROS2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\e9_Ros2Install

文件夹/文件名称	说明
install-ROS2-ubuntu.sh	安装配置文件
RosSwitch.bat	ROS环境切换脚本
WslGUI.bat	GUI窗口启动脚本
ros_switch.sh	ros环境切换配置文件

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链。

> ①：若使用Pixhawk 6X飞控，平台安装时的编译命令为：px4_fmu-v6x_default，推荐PX4固件版本为：1.12.3。其他配套飞控及编译命令请见：https://rflysim.com/doc/zh/1/Hardware.html

4.2 硬件要求

笔记本/台式电脑① 1台。

> ①：推荐配置请见：https://rflysim.com/doc/zh/HowToInstall.pdf

5.实验步骤

Ros2环境的安装与配置

首先，我们需要一台装有Ubuntu的电脑或虚拟机，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-ROS2-ubuntu.sh脚本，运行命令./install-ROS2-ubuntu.sh，运行命令后，便会进行Ros2环境的安装。

在安装完成之后，需要输入如下命令：

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

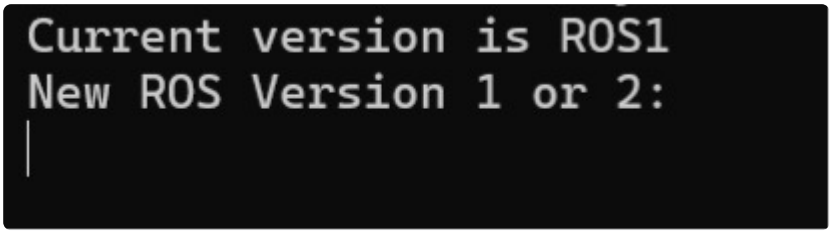
注：gedit也可以换成vim。

如果在安装ROS2之前，已经安装过ROS1，并希望ROS1和ROS2并存。则删除原来的ROS1条目中的“source /opt/ros/noetic/setup.bash”替换成下列中的语句：

```
ros_version=1 if [ $ros_version -eq 1 ]; then source /opt/ros/noetic/setup.bash else source /opt/ros/foxy/setup.bash fi
```

之后保存并退出。

然后，再拷贝过来的文件夹中按下CTRL+ALT+T打开一个新的终端。运行ros_switch.sh文件，该文件可以用来切换ROS1和ROS2的环境。切换ROS1环境的命令为“~/ros_switch.sh 1”；切换ROS2的环境的命令为“~/ros_switch.sh 2”；或者输入“~/ros_switch.sh”，并根据提示来切换ROS1或ROS2环境。



如果后续只想要使用ROS2，则在删除原来ROS1的条目，新添加语句：

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

之后，在终端中输入：

```
source ~/.bashrc
```

这样，就能加载ROS2的环境，后续ROS2环境就能自动加载了。

注：ROS1是noetic，而ROS2是foxy。

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ROS2开发环境以及同时部署ROS1与ROS2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。

ROS2的虚拟机链接:

https://pan.baidu.com/s/1u5VuFIeMsqm-boj9J8Ug-g?pwd=h25i 提取码: h25i

配置ROS1与ROS2的虚拟机链接:

https://pan.baidu.com/s/1PALGty97x2YOvX4D3ilTqw?pwd=g8nk 提取码: g8nk

以上虚拟机的账户名以及密码均为nvidia。

Ros2环境的测试

Ubuntu下的环境测试：

Step1：启动ROS2环境

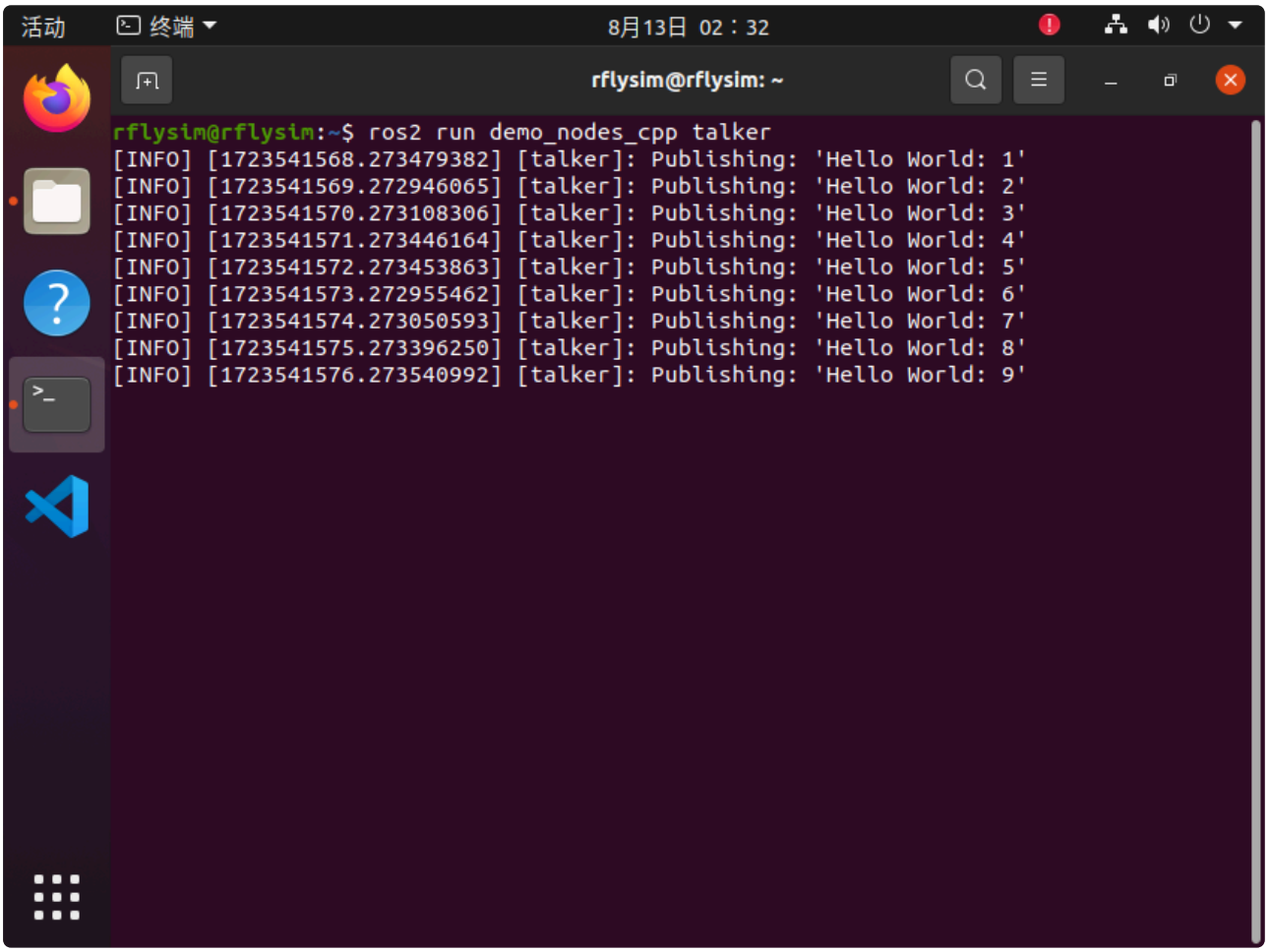
打开一个新的终端窗口。

如果你已经按照ROS2的安装指南正确设置了环境，那么你应该能够在终端中直接运行ROS2命令。

Step2：运行ROS2节点

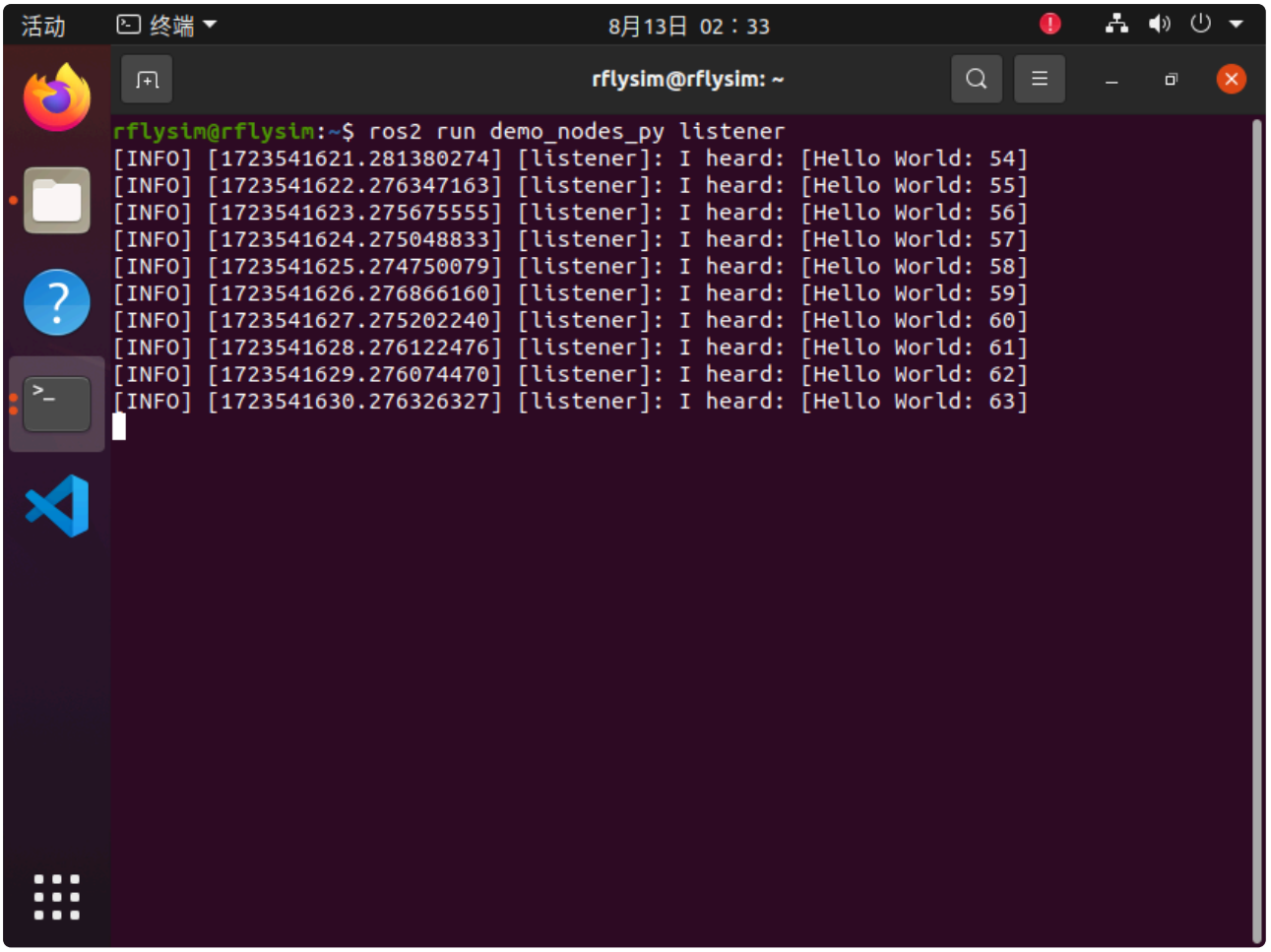
输入下面的命令来运行一个发布者节点。这个节点将会发布一个话题并持续发送消息。

```
ros2 run demo_nodes_cpp talker
```



打开另一个终端窗口，并输入下面的命令来运行一个订阅者节点。这个节点将会订阅之前发布者节点发布的话题，并打印接收到的消息。

```
ros2 run demo_nodes_py listener
```

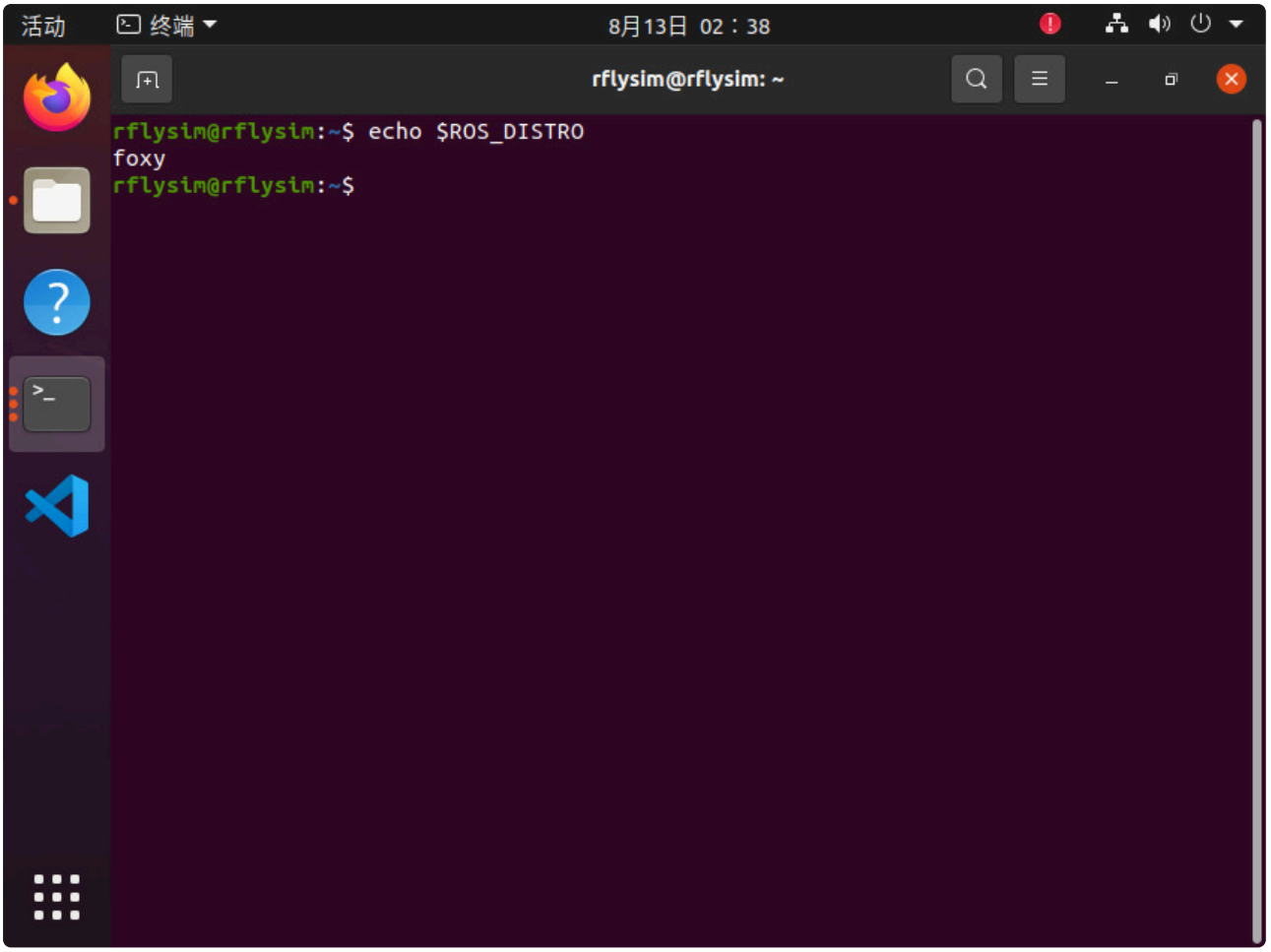


如果你在订阅者节点的终端窗口中看到了来自发布者节点的消息，那么说明ROS2环境已经正确安装并且可以正常工作了。

Step3：检查ROS2环境变量

输入下面的命令来检查你的ROS2发行版环境变量是否已设置。这将显示你当前使用的ROS2发行版名称，如foxy、galactic等。

```
echo $ROS_DISTRO
```

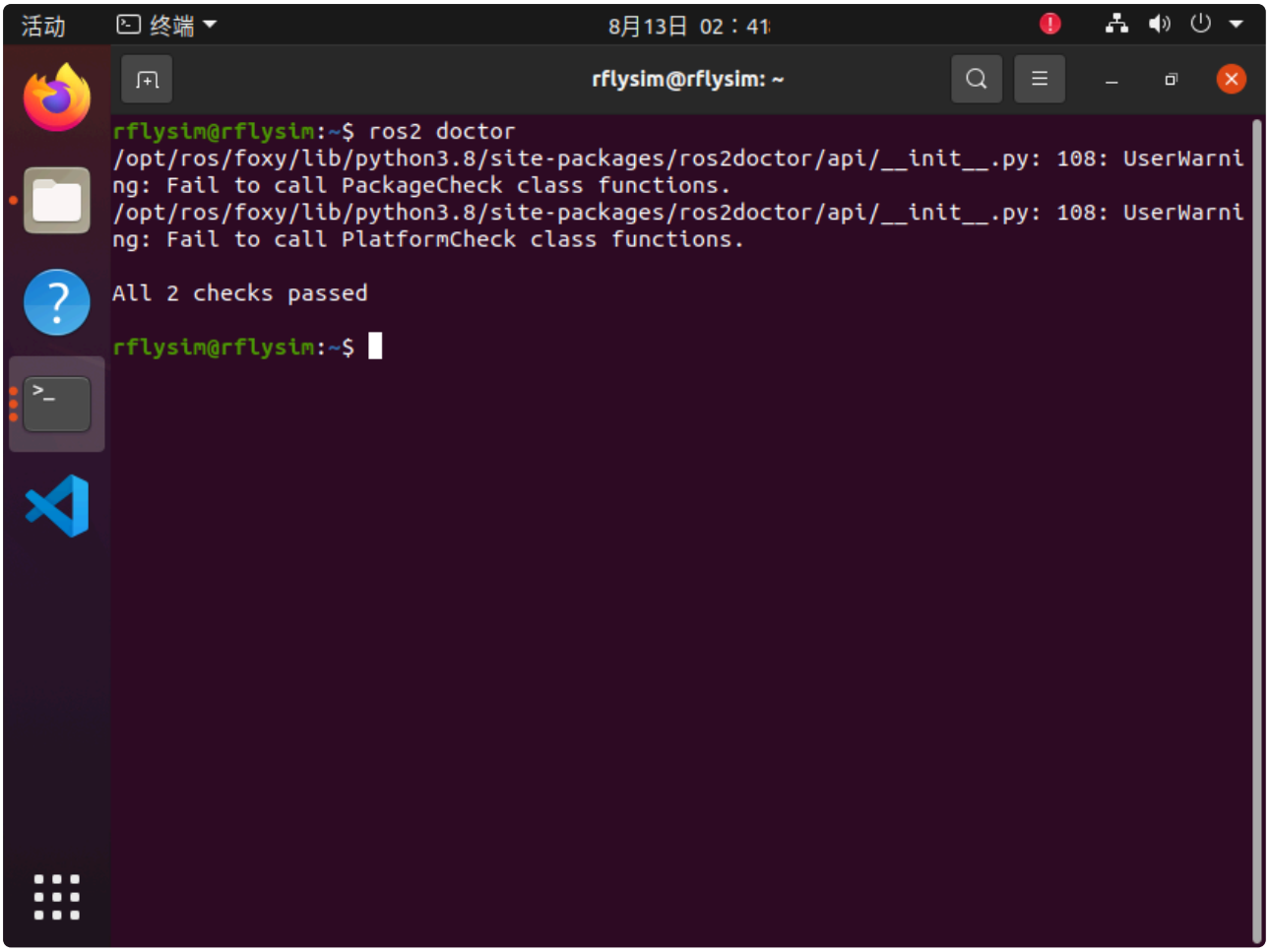


同时，你也可以检查其他与ROS2相关的环境变量，如 ROS_DOMAIN_ID、ROS_MASTER_URI
等，以确保它们已被正确设置。

Step4：使用ros2 doctor命令进行诊断

ROS2提供了一个名为ros2
doctor的命令行工具，用于诊断ROS2系统中的问题。在终端中输入下面的命令并查看输出信息，以确保没有错误或警告。

```
ros2 doctor
```



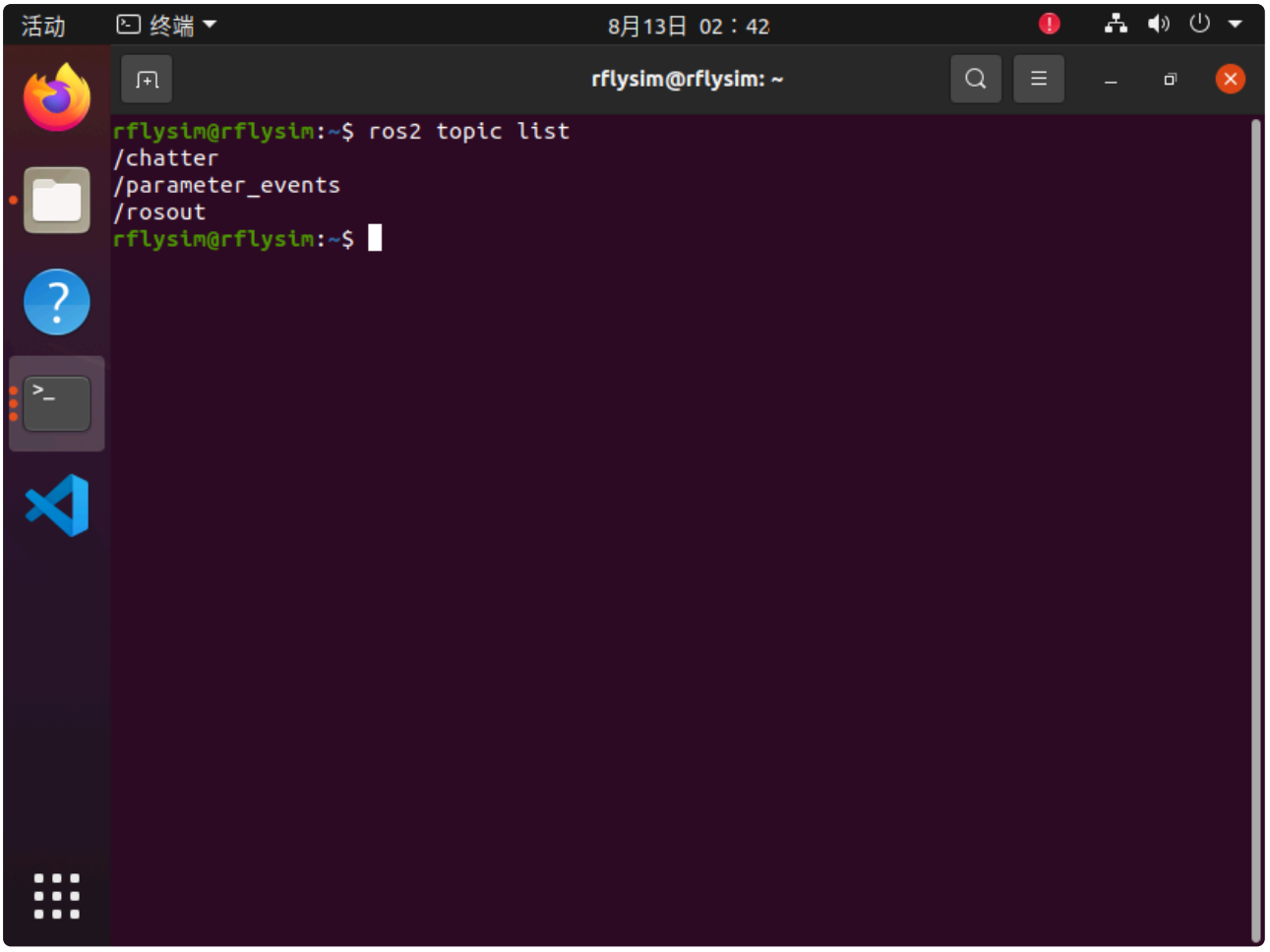
Step5：查看ROS2话题和节点

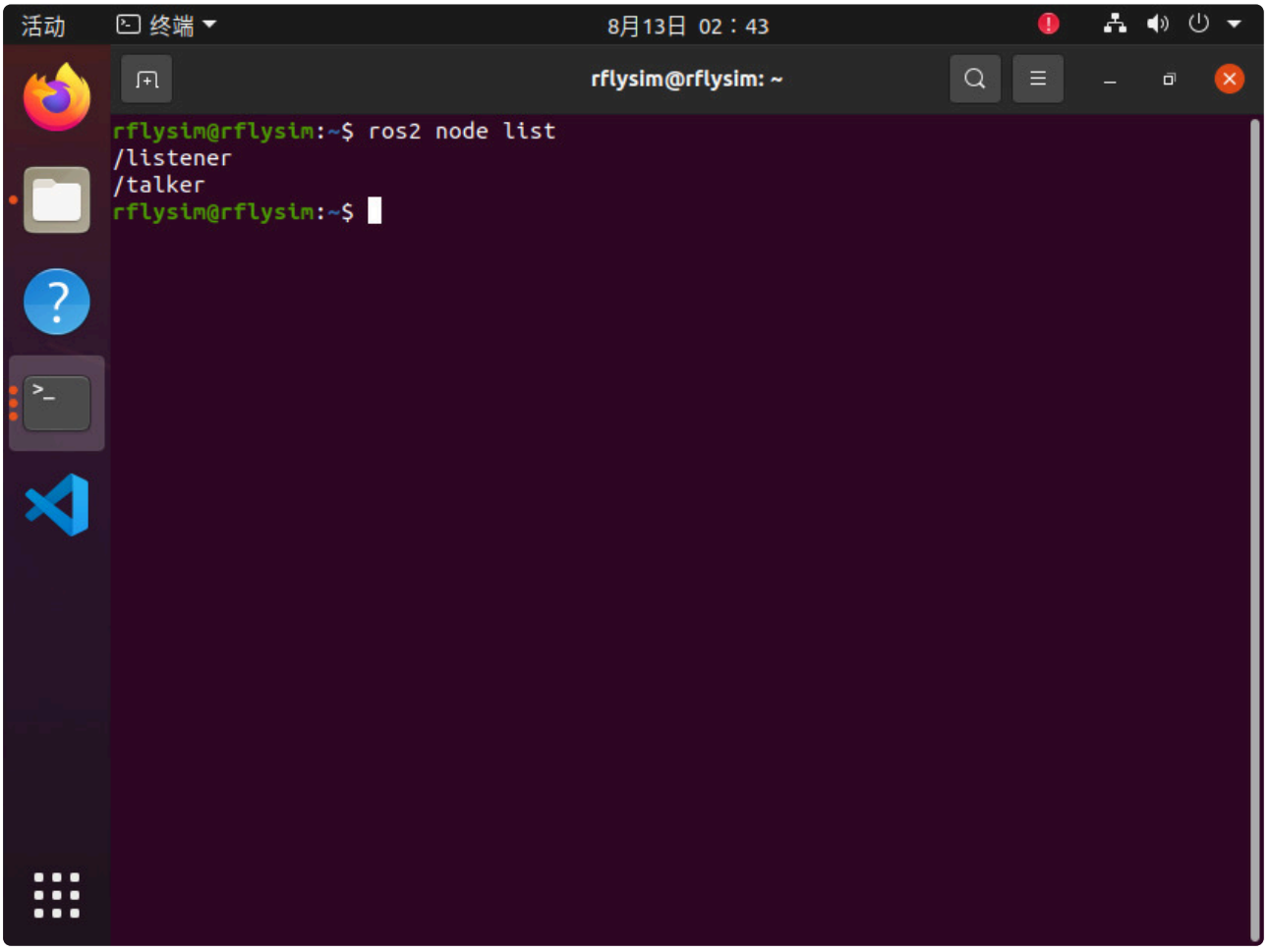
使用下面的命令来查看当前运行的话题列表。

```
ros2 topic list
```

使用下面的命令来查看当前运行的节点列表。

```
ros2 node list
```


活动
终端
8月13日 02：42
rflysim@rflysim: ~
rflysim@rflysim:~$ ros2 topic list
/chatter
/parameter_events
/rosout
rflysim@rflysim:~$


活动
终端
8月13日 02：43
rflysim@rflysim: ~
rflysim@rflysim:~$ ros2 node list
/listener
/talker
rflysim@rflysim:~$

这些命令将帮助你确认ROS2系统中的各个组件是否正在正常运行。

如果你完成了以上步骤并且没有遇到问题，那么你的ROS2环境应该已经正确安装了。

WinWSL下的环境测试：

1. 运行Desktop\RflyTools\WslGUI快捷方式或文件夹“WslGUI.bat”，打开一个GUI窗口。

2. 在其中，按下CTRL+ALT+T的快捷键，打开一个终端。输入

ros2 run demo_nodes_cpp talker

启动一个数据的发布者节点。

3. 按下CTRL+ALT+T的快捷键，打开一个新终端。输入

ros2 run demo_nodes_py listener

启动一个数据的订阅者节点。

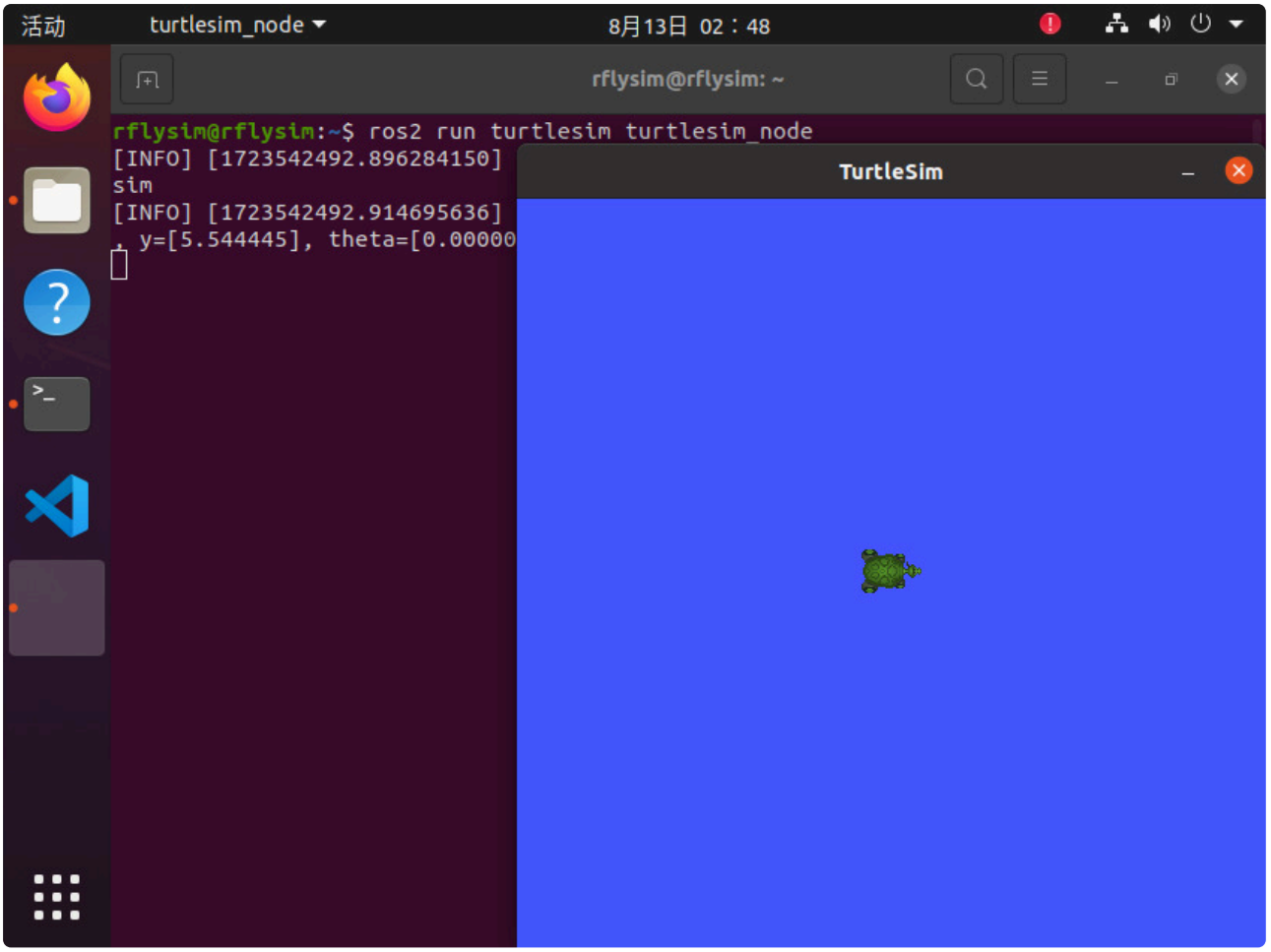
4. 如果“Hello World”字符串在两个终端中正常传输，说明通信系统没有问题。

经典海龟控制demo

Step1：

在Ubuntu系统下，按下CTRL+ALT+T的快捷键，打开一个终端。输入下面的命令来启动小海龟仿真器：

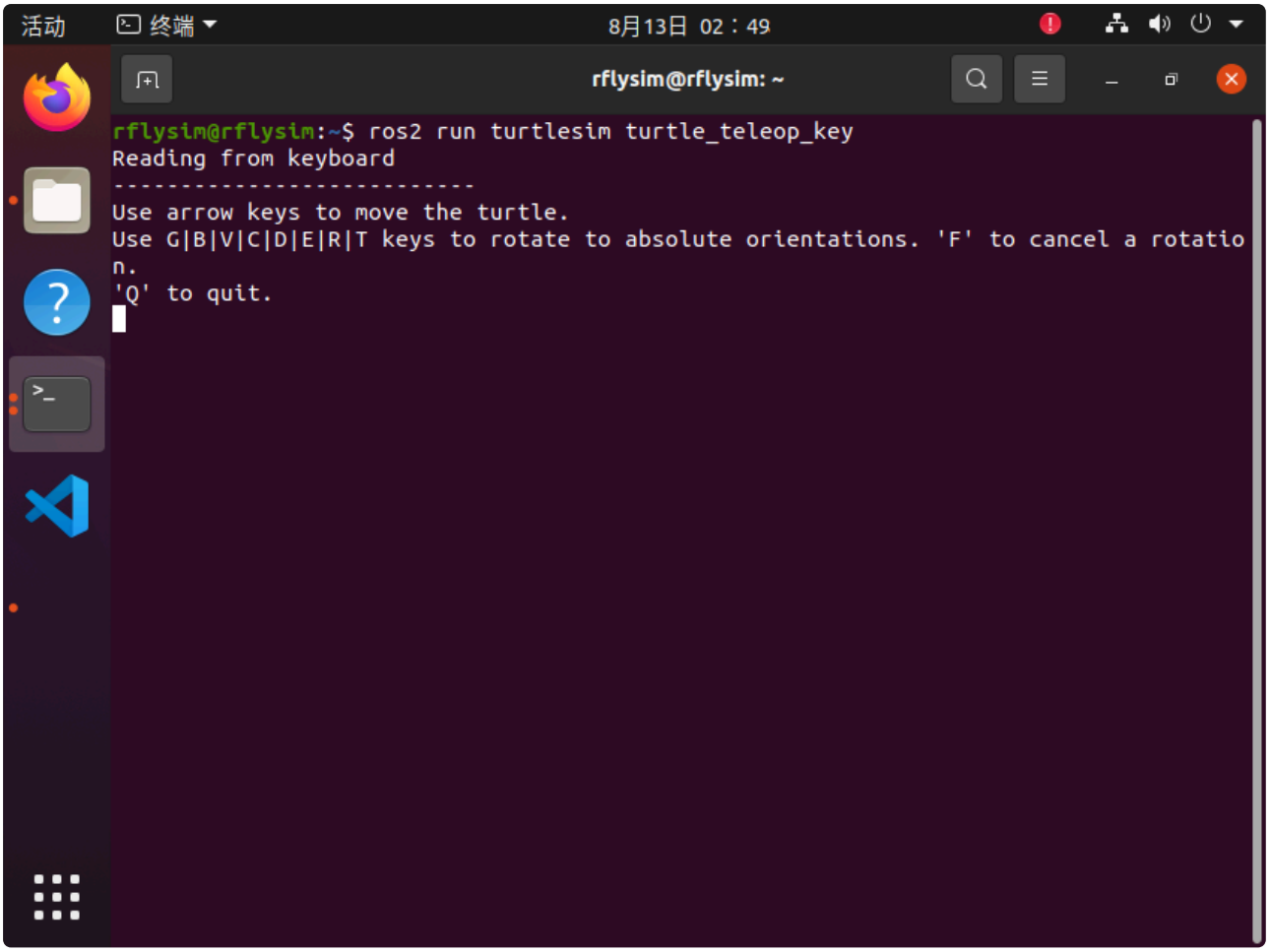
```
ros2 run turtlesim turtlesim_node
```



Step2：

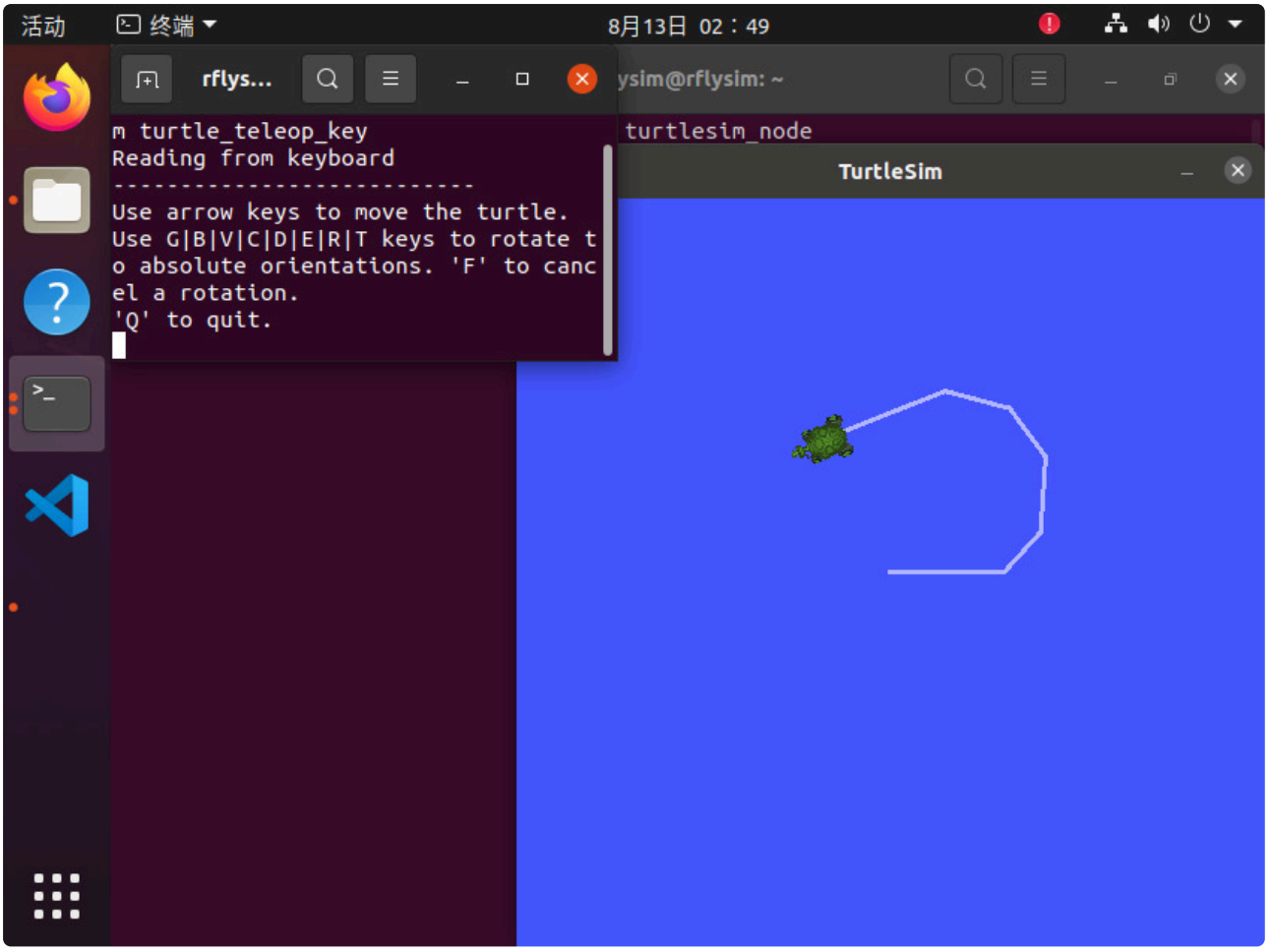
再次按下CTRL+ALT+T的快捷键，打开一个新的终端。输入下面的命令启动海龟键盘控制节点：

```
ros2 run turtlesim turtle_teleop_key
```



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ROS2，则切换到ROS2。

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：

ros2 run turtlesim turtlesim_node

来启动小海龟仿真器。

4. 按下CTRL+ALT+T的快捷键，打开一个新终端。输入

ros2 run 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）

注：ROS2的大部分命令，都是遵循下面的迁移原则

ros*** -> ros2 ***

例如：

rosrun -> ros2 run

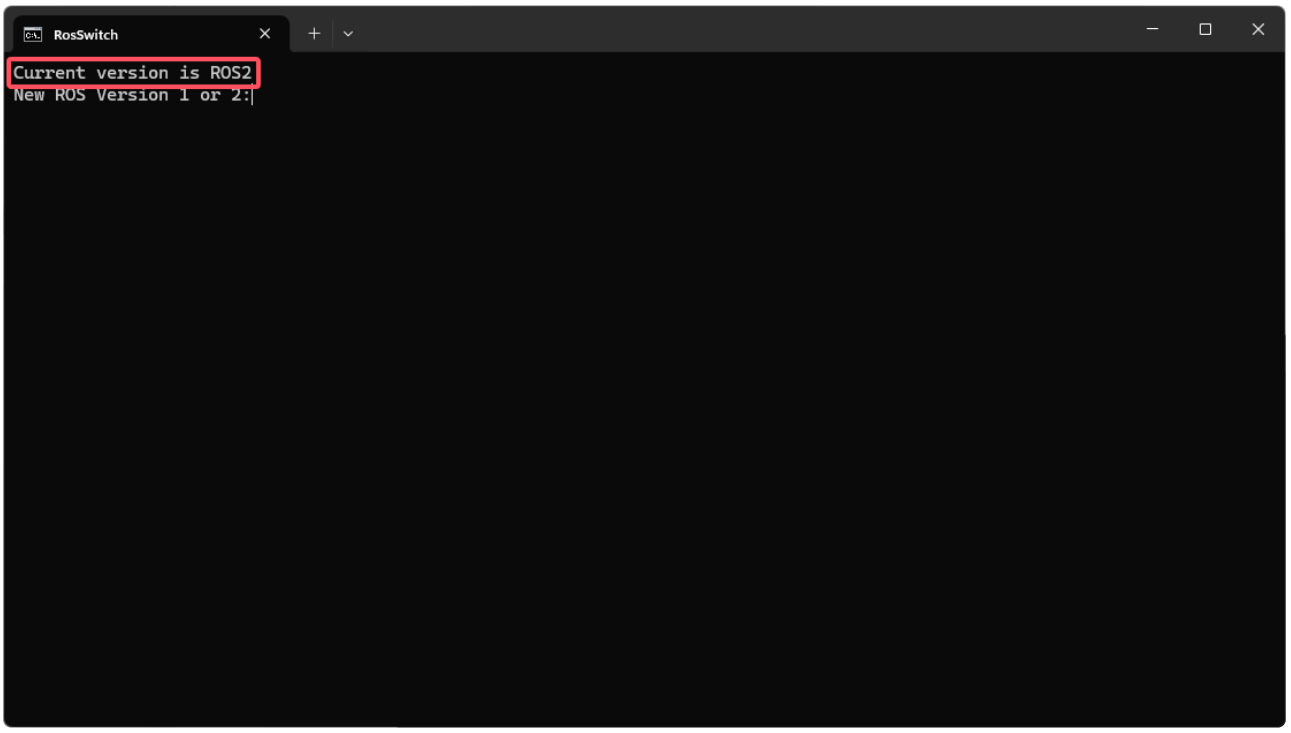
rostopic -> ros2 topic

roslaunch -> ros2 launch

ROS2例程C++版（WinWSL，必做）

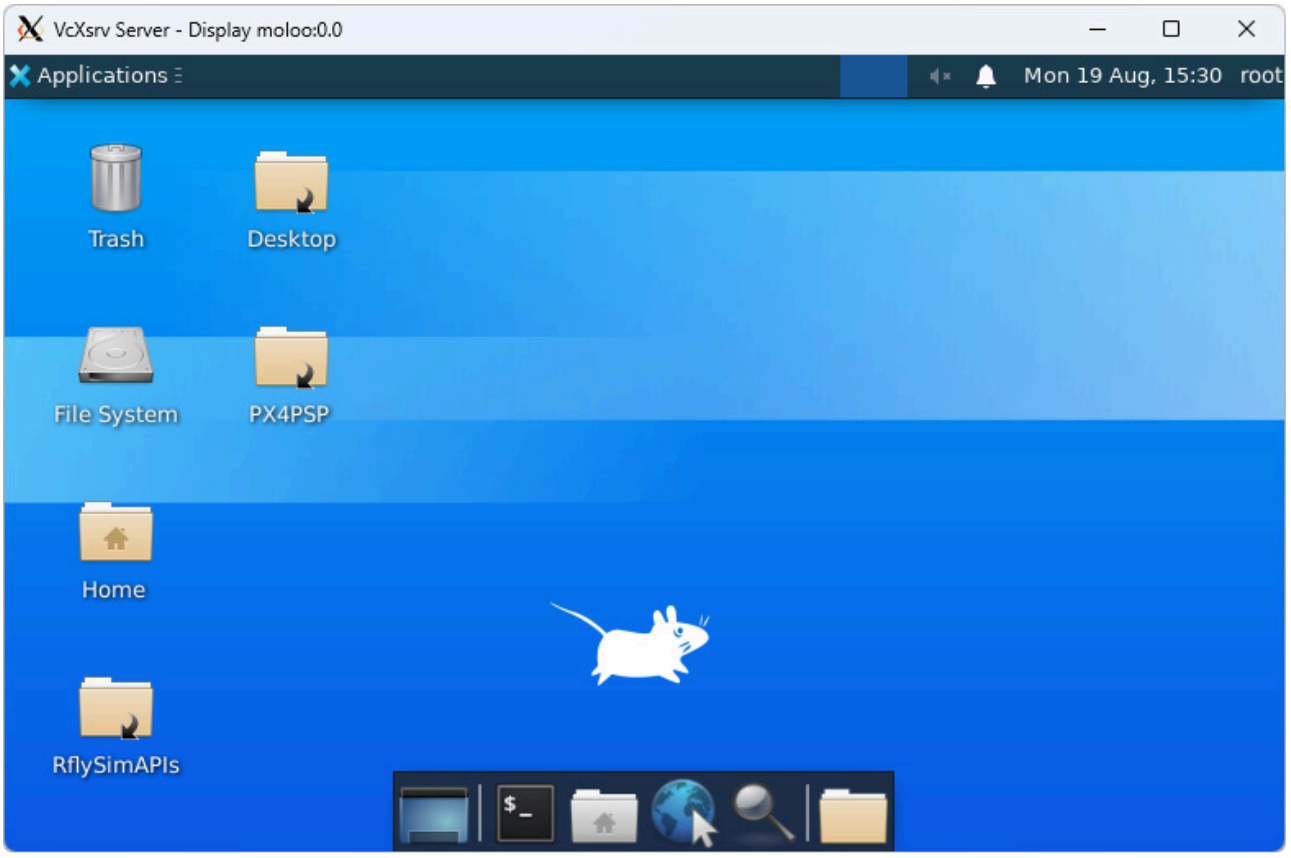
Step1：

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ROS2，如果不是ROS2，请输入2并按回车，切换为ROS2环境。



Step2：

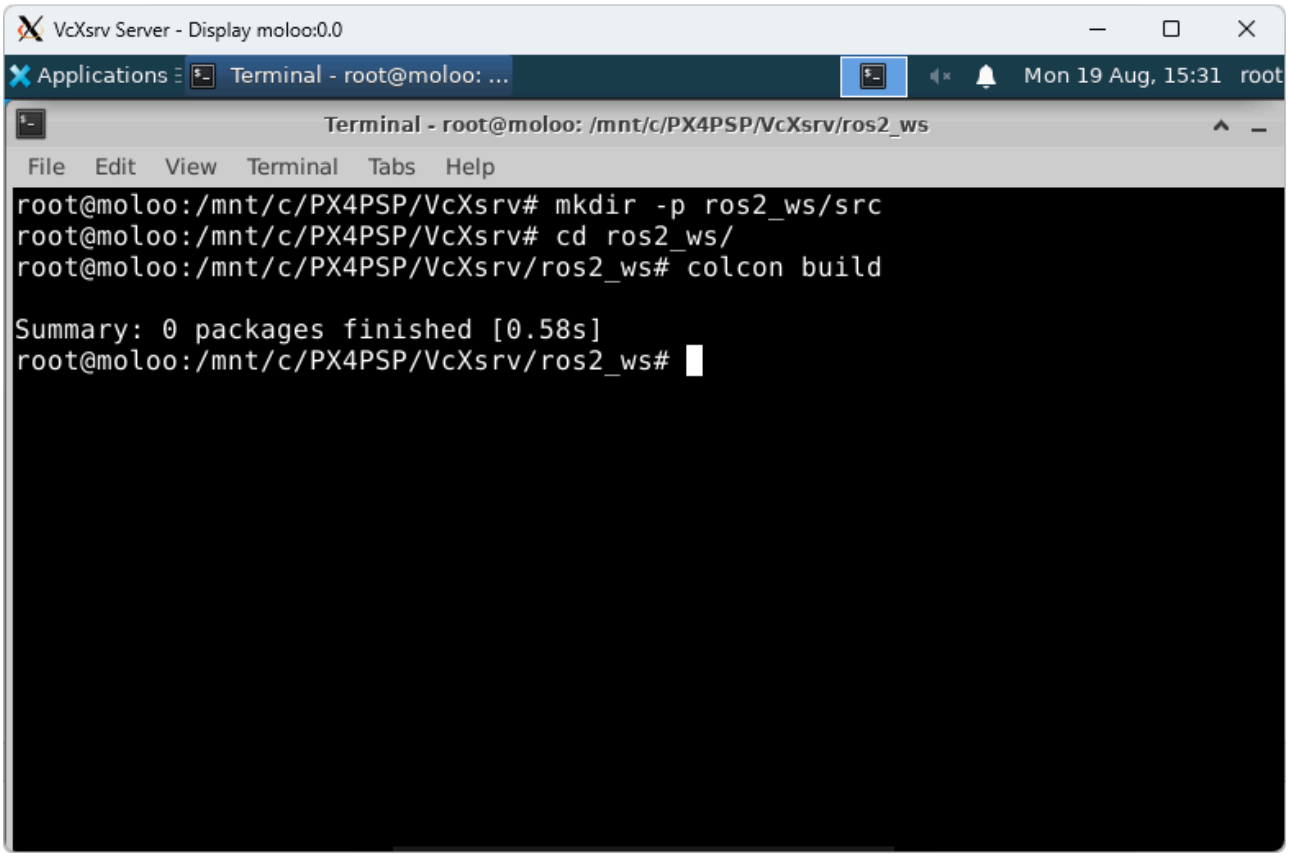
打开“桌面\RflyTools\WslGUI”，为WinWSL的图形化界面。



Step3：

创建ros2工作空间。

mkdir -p ros2_ws/src cd ros2_ws colcon build # 编译全部节点



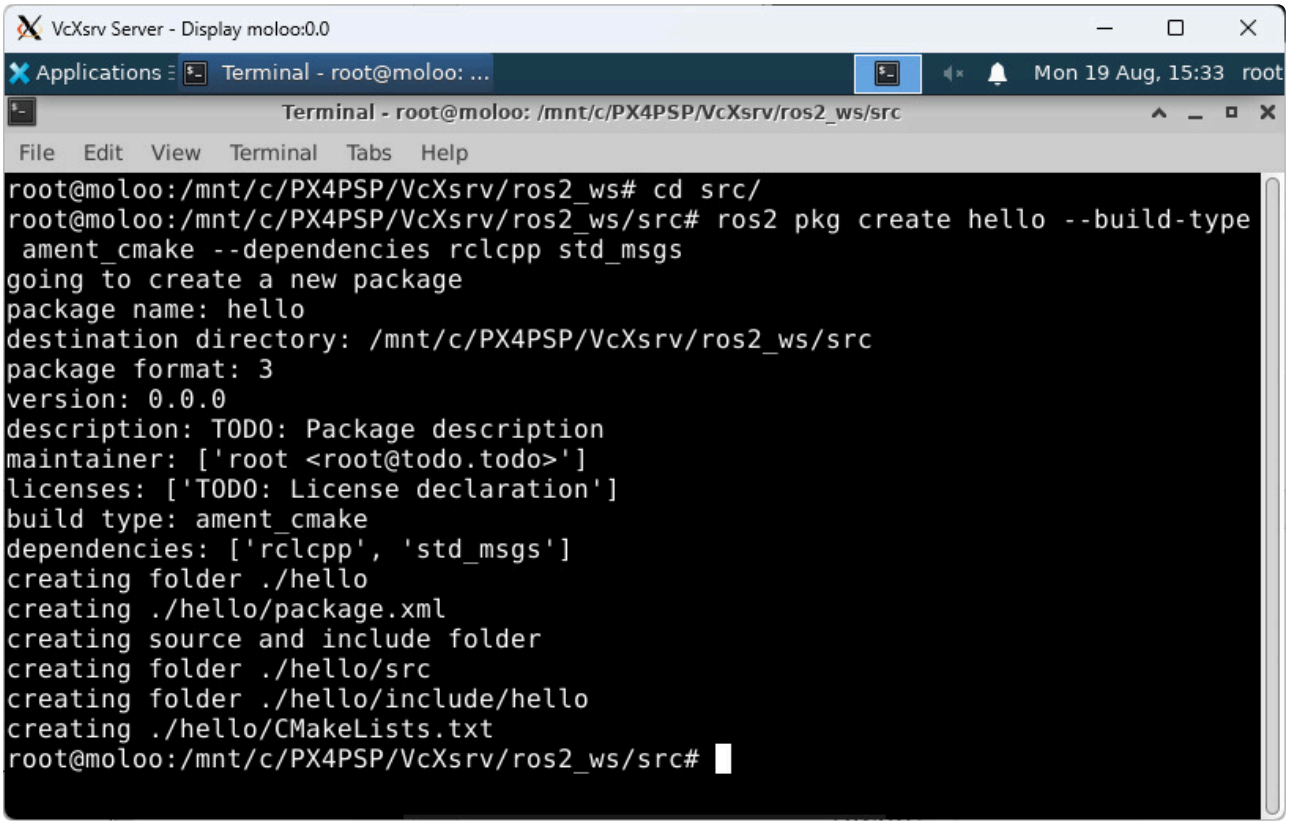
Step4：

创建功能包。

cd src # 创建C/C++功能包 命令行 # 注意： # ament_cmake 使用 C/C++ 的功能包 # ament_python 使用 Python 功能包 ros2 pkg create node_name --build-type ament_cmake --dependencies rclcpp std_msgs

这里我们命名为hello，所以代码修改如下。

ros2 pkg create hello --build-type ament_cmake --dependencies rclcpp std_msgs



Step5：

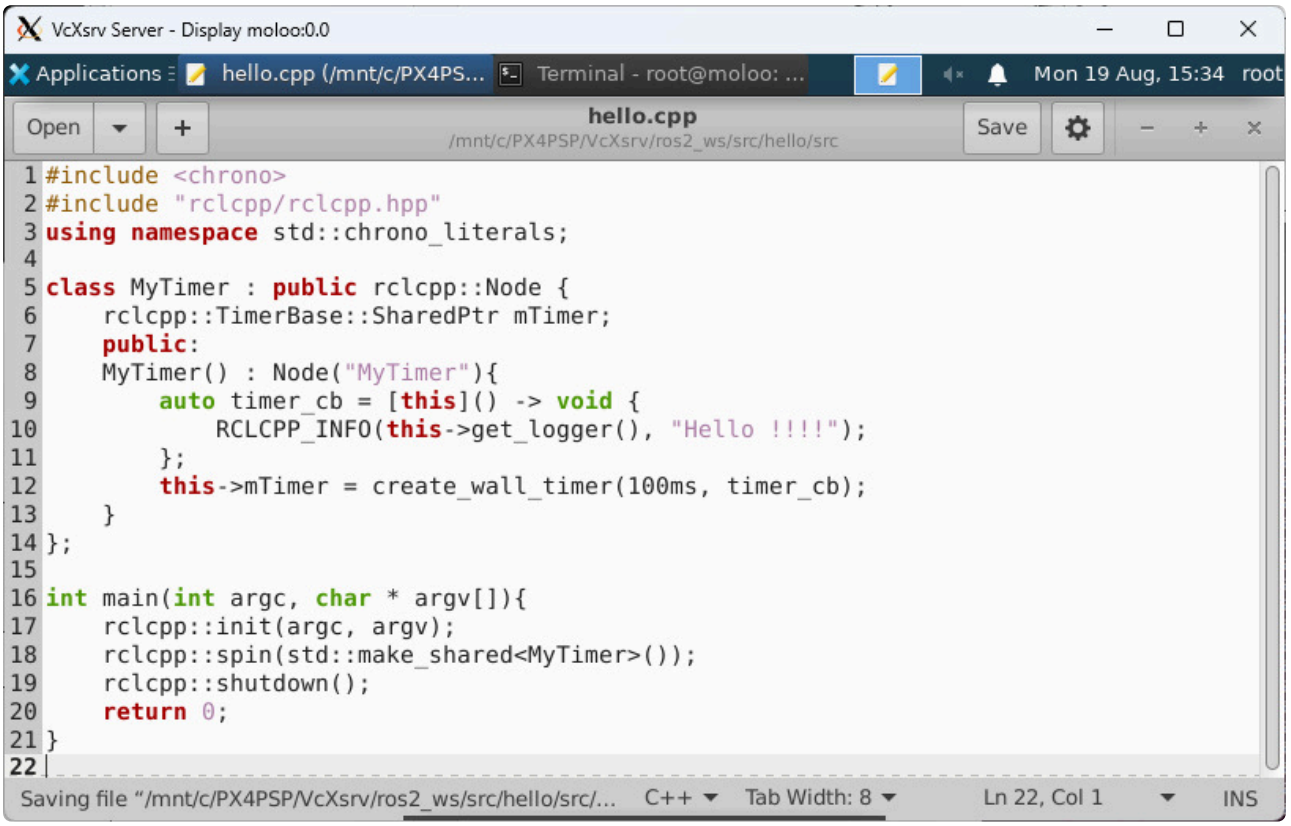
创建节点源文件。

注：也可以在创建包的命令中，通过添加“--node-name node_name”选项，来自动创建节点，免去手动创建的过程。

 # 进入 src 目录 cd hello/src/ touch hello.cpp

使用gedit编写代码，复制进如下的内容。

```
#include <chrono> #include "rclcpp/rclcpp.hpp" using namespace std::chrono_literals; class MyTimer : public rclcpp::Node { rclcpp::TimerBase::SharedPtr mTimer; public: MyTimer() : Node("MyTimer"){ auto timer_cb = this -> void { RCLCPP_INFO(this->get_logger(), "Hello !!!!"); }; this->mTimer = create_wall_timer(100ms, timer_cb); } }; int main(int argc, char * argv[]){ rclcpp::init(argc, argv); rclcpp::spin(std::make_shared<MyTimer>()); rclcpp::shutdown(); return 0; }
```



Step6：

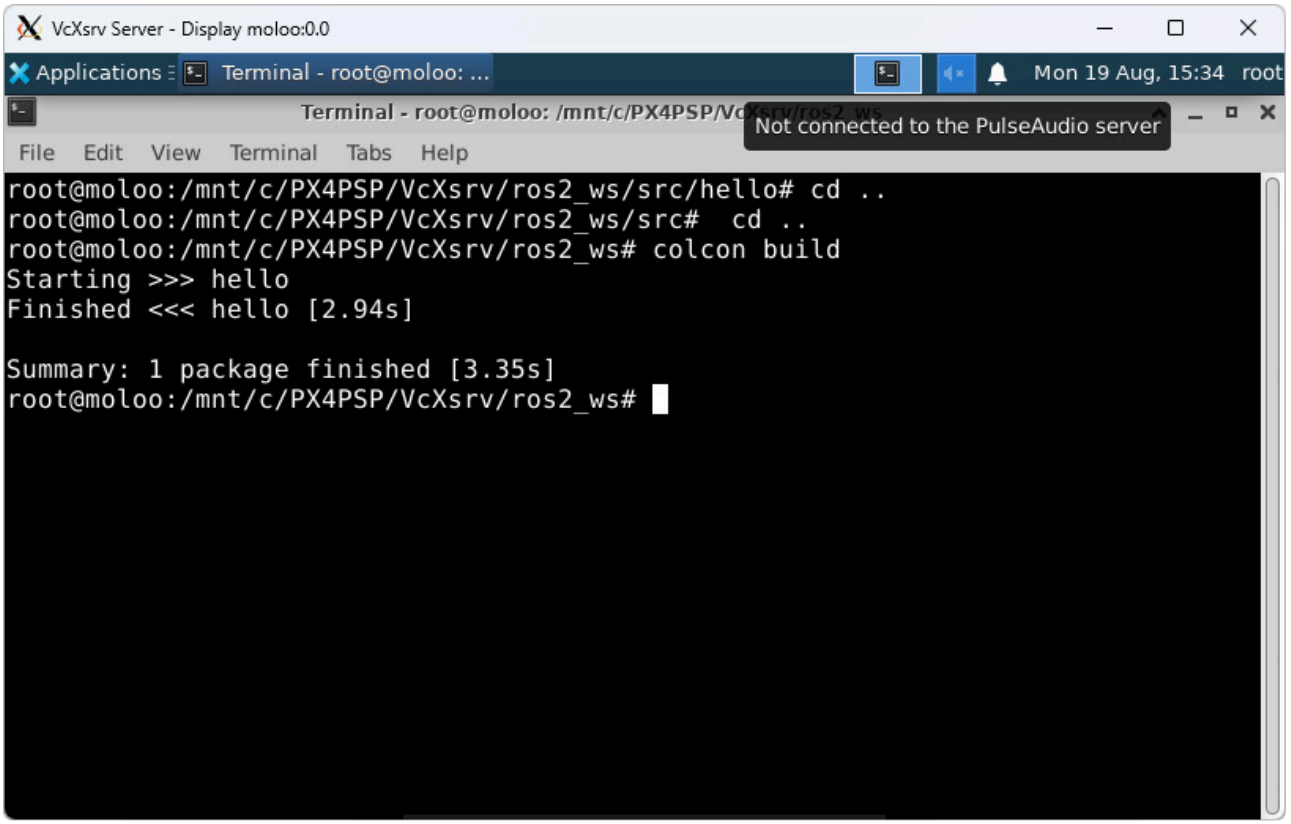
编写 CMake 脚本。

```
cd .. # 1.打开 CMakeLists.txt gedit CMakeLists.txt # 2.增加节点编译脚本 add_executable(hello src/hello.cpp) ament_target_dependencies(hello rclcpp std_msgs) # 3.增加安装脚本 install(TARGETS hello DESTINATION lib/hello)
```

Step7：

编译 ROS2 程序。

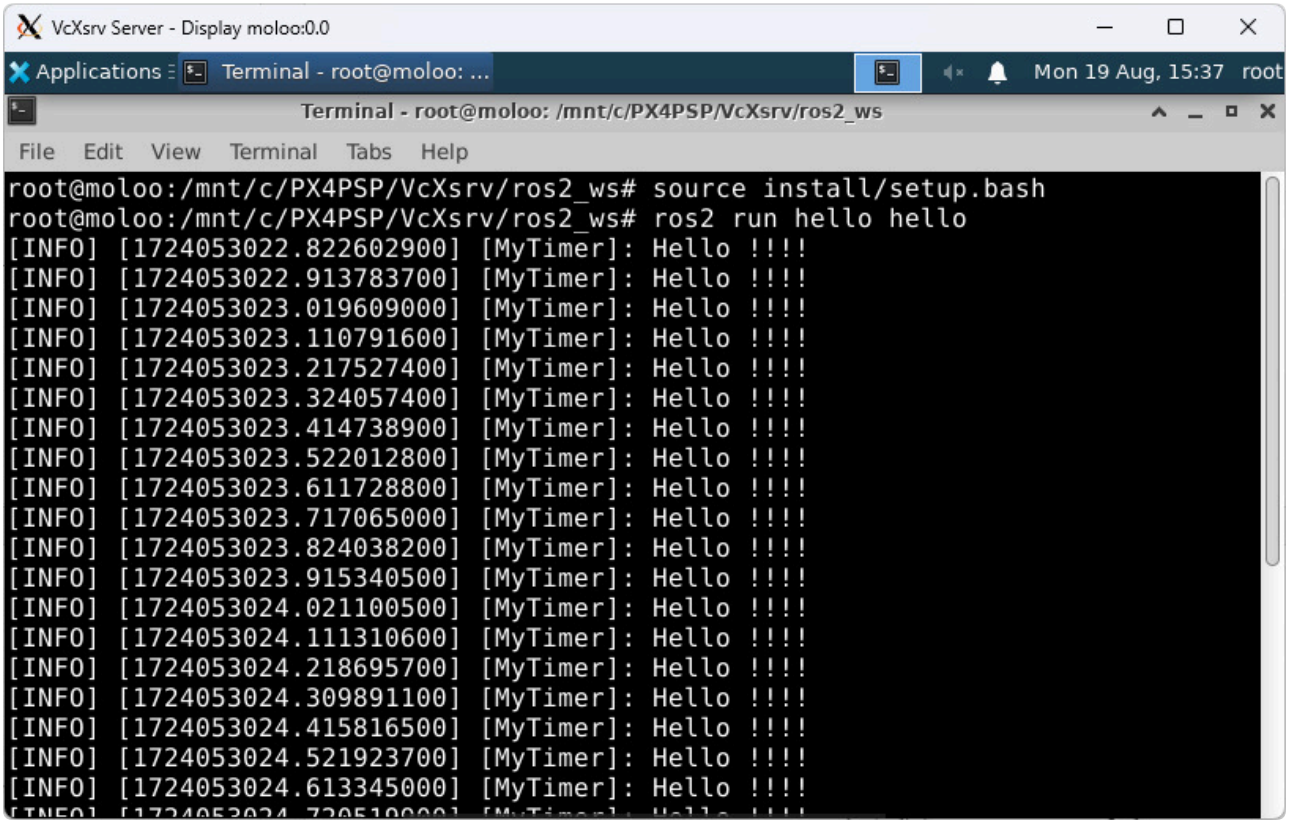
```
# 1.进入根目录 cd .. cd .. # 2.编译程序 colcon build --packages-select hello # 编译指定节点 colcon build # 编译全部节点
```



Step8：

运行 ROS2 程序。

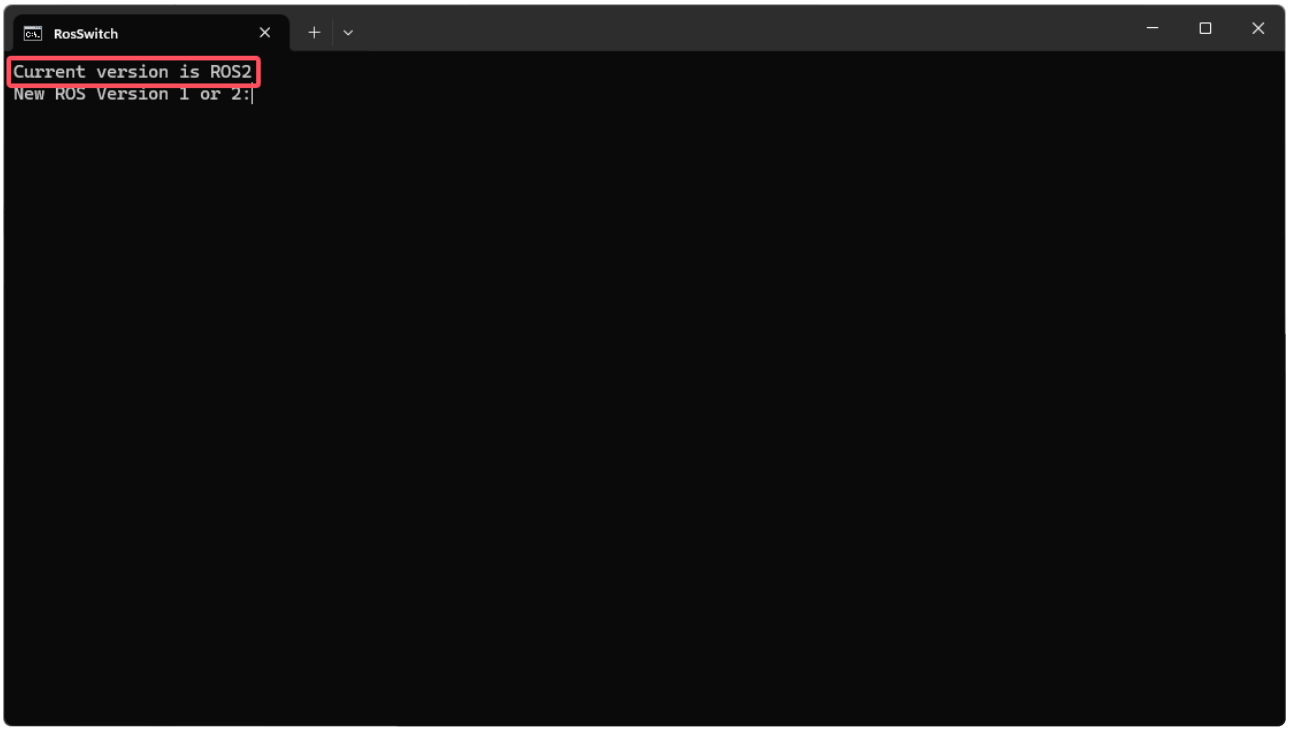
 # 1.加载环境 source install/setup.bash # 2.运行程序 ros2 run hello hello



ROS2例程python版（WinWSL，必做）

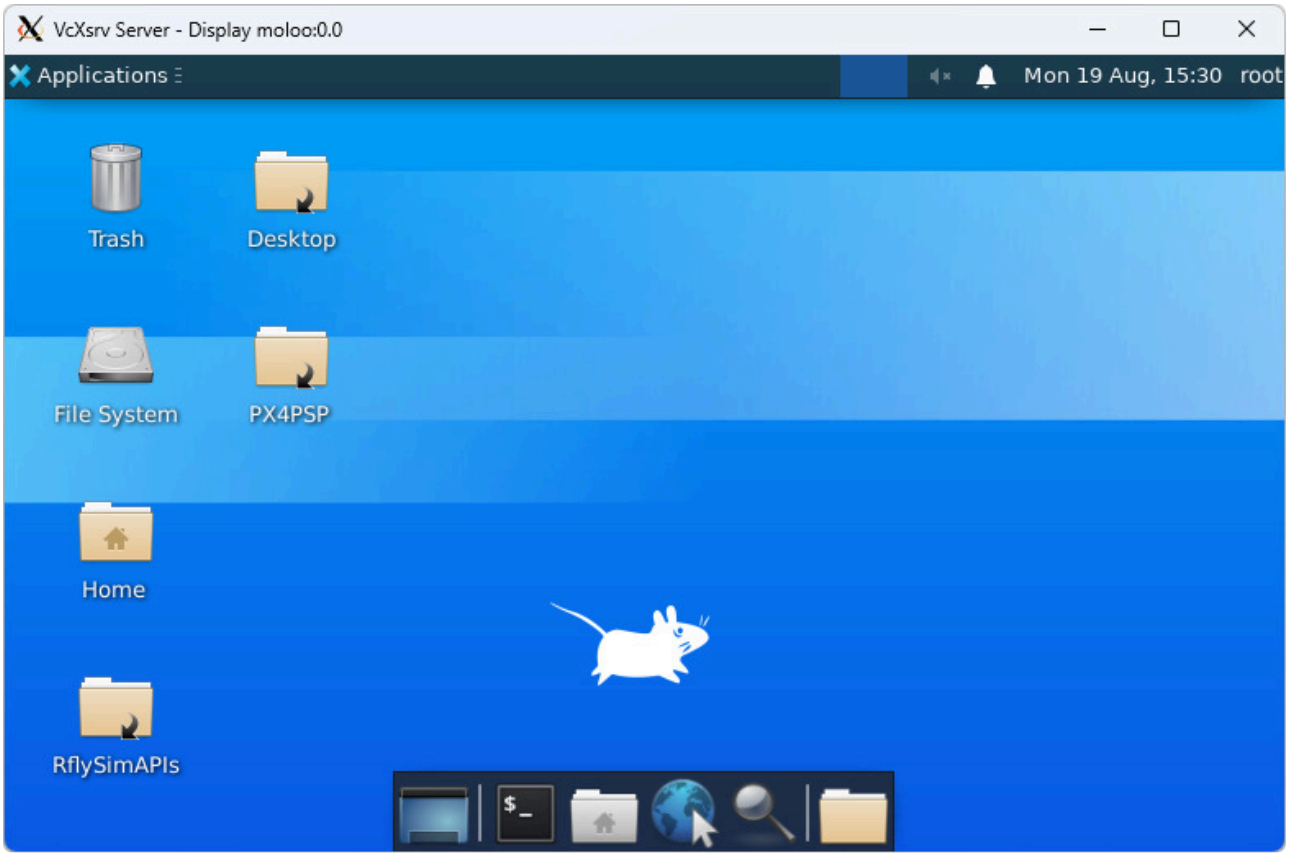
Step1：

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ROS2，如果不是ROS2，请输入2并按回车，切换为ROS2环境。



Step2：

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



Step1：

如果按照上一节的内容，会创建好工作空间，无需继续创建。如果您还没有创建工作空间，请创建ros2工作空间。

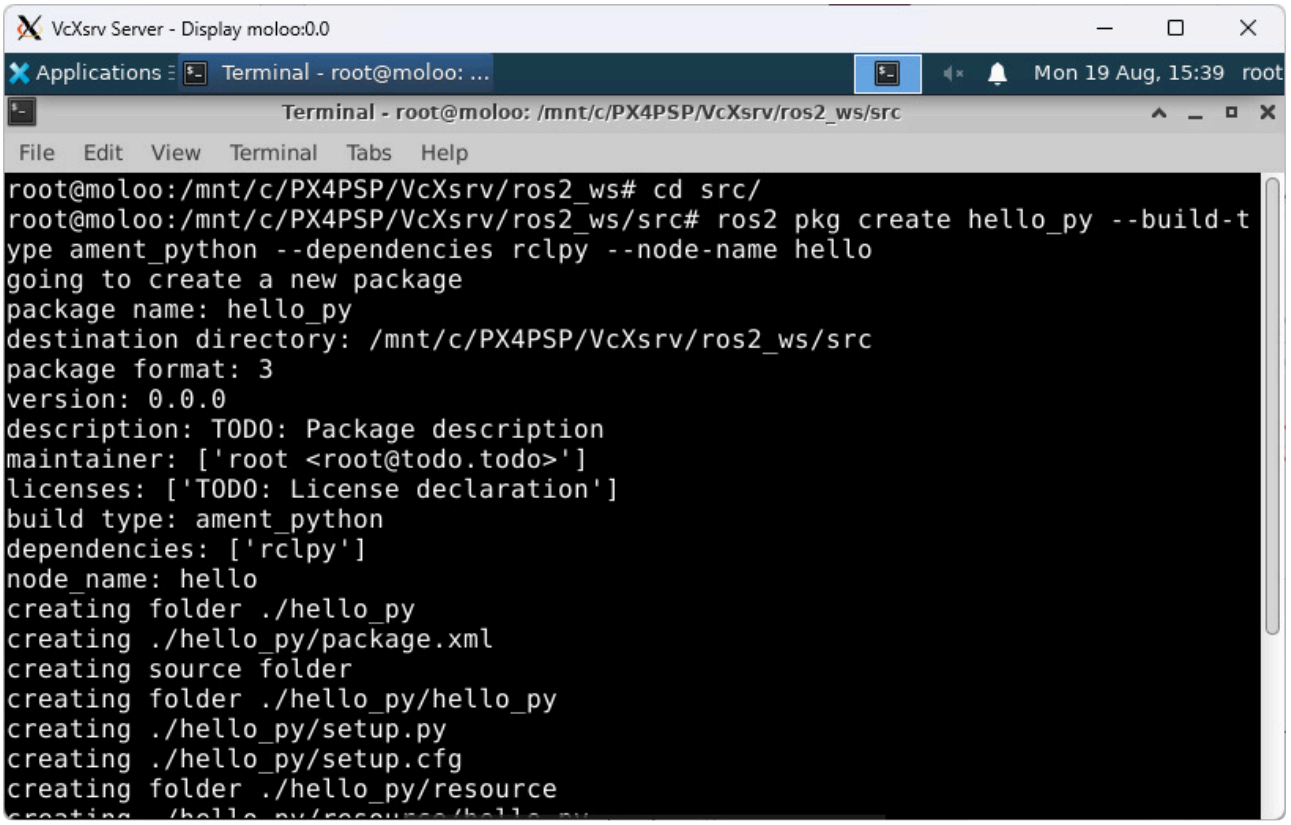
```
mkdir -p ros2_ws/src cd ros2_ws colcon build # 编译全部节点
```

Step2：

进入工作空间，创建功能包和节点源文件。包名为hello_py，节点名为hello。

注：下面的命令，既可以创建功能包，还可以创建节点源文件，免去手动创建节点源文件的过程。C++的代码也可以通过添加“--node-name node_name”选项，来创建节点。

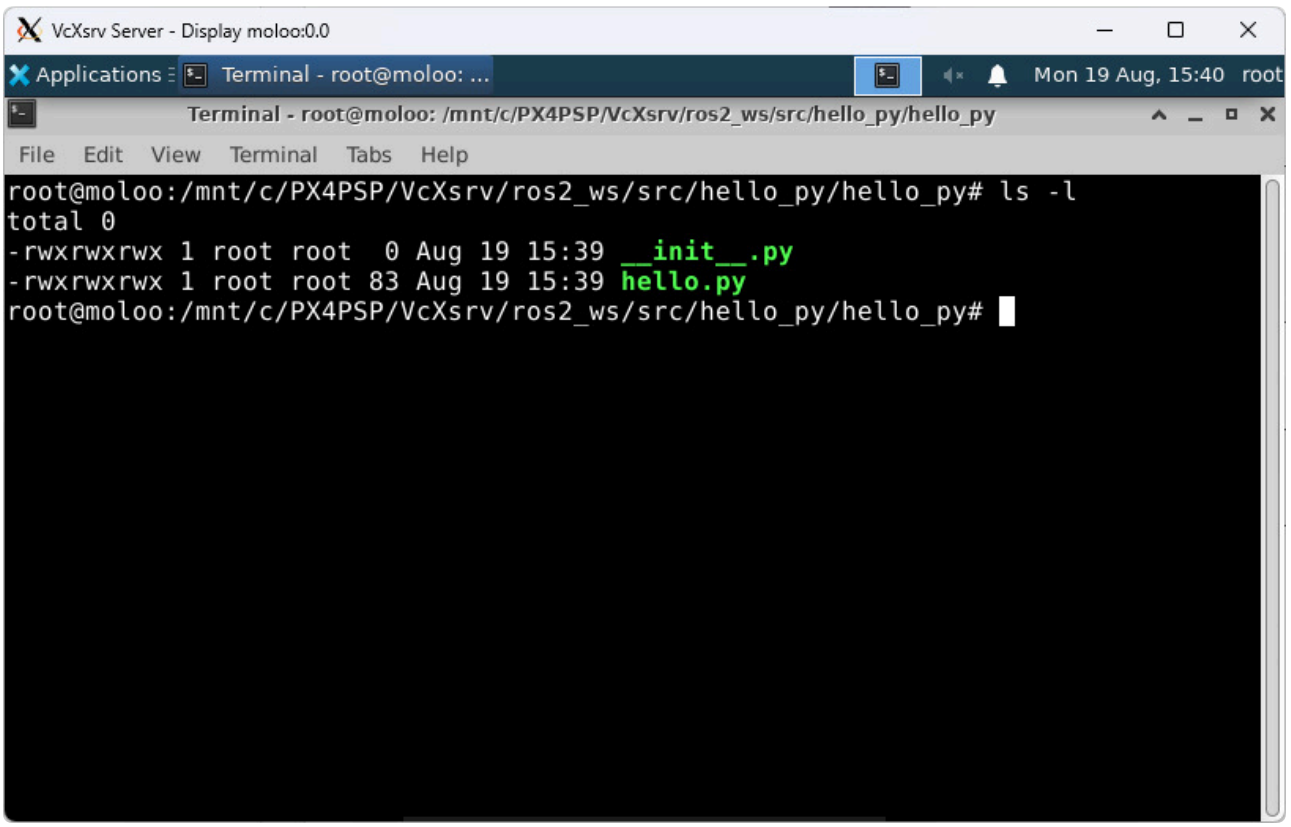
```
cd src # 创建python功能包 命令行 # 注意： # ament_cmake 使用 C/C++ 的功能包 # ament_python 使用 Python 功能包 ros2 pkg create hello_py --build-type ament_python --dependencies rclpy --node-name hello
```



Step3：

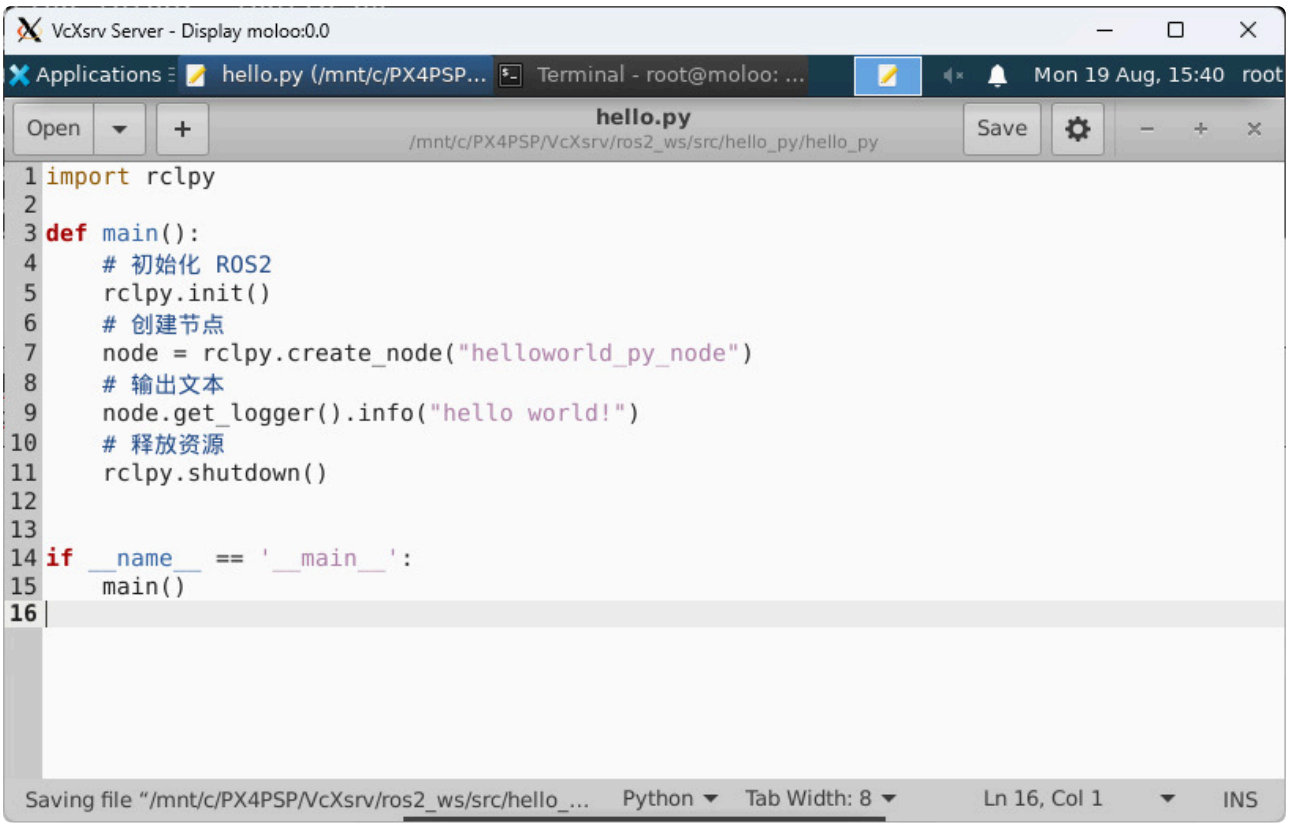
节点源文件已经创建好，直接编辑源文件。

cd hello_py/hello_py # 进入该目录后，应该已经有hello.py文件，否则需要手动创建
gedit hello.py



将下面代码复制进去，并保存。

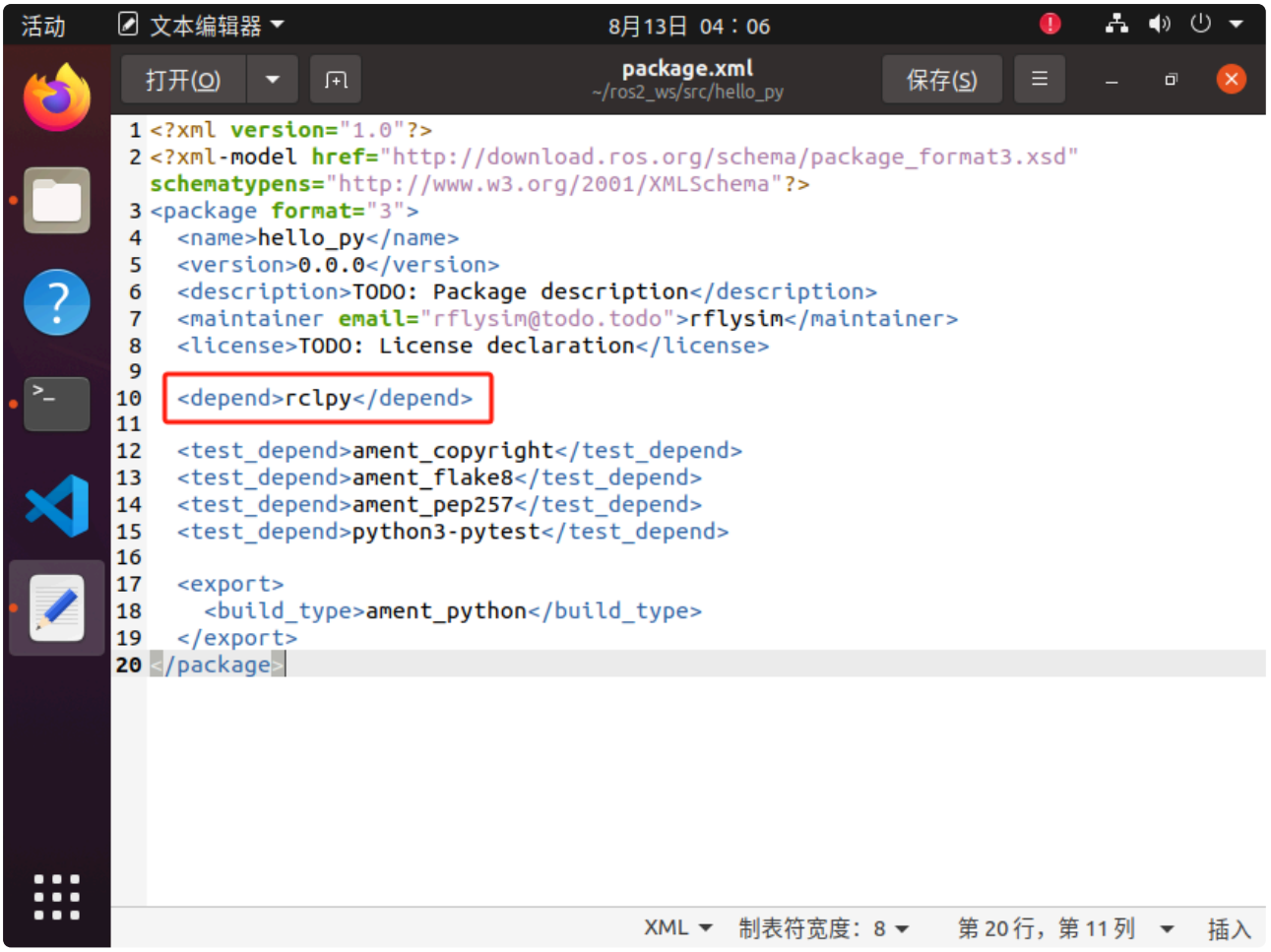
import rclpy def main(): # 初始化 ROS2 rclpy.init() # 创建节点 node = rclpy.create_node("helloworld_py_node") # 输出文本 node.get_logger().info("hello world!") # 释放资源 rclpy.shutdown() if *name*_ == 'main': main()



Step3：

编辑配置文件。与C++类似的，在步骤1创建功能包时所使用的指令也已经默认生成且配置了配置文件，不过实际应用中经常需要自己编辑配置文件，所以在此对相关内容做简单介绍，所使用的配置文件主要有两个，分别是功能包下的package.xml与setup.py。

文件package.xml的内容如下，红色方框标记的内容可根据需要添加、删除和修改。

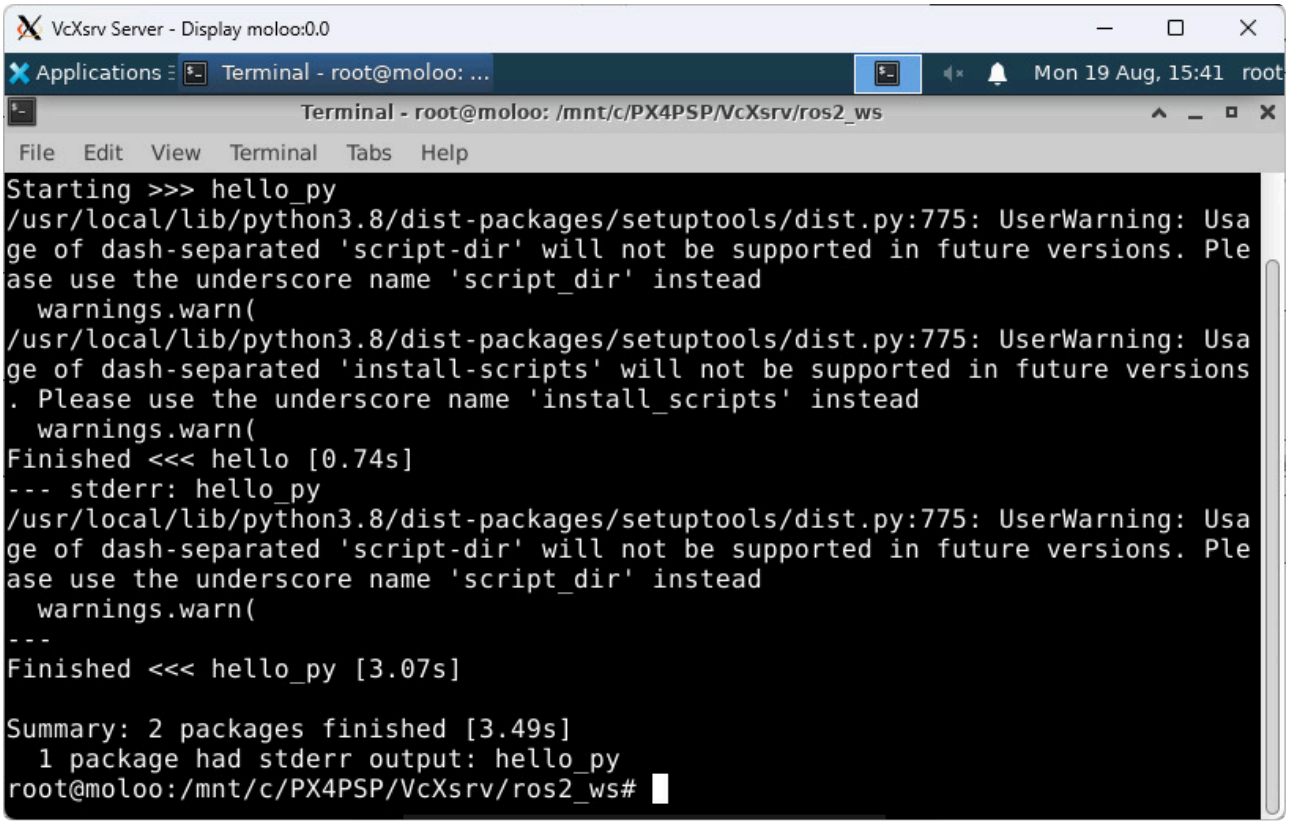


文件setup.py的内容如下，红色方框标记的内容可根据需要修改。

Step4：

编译。

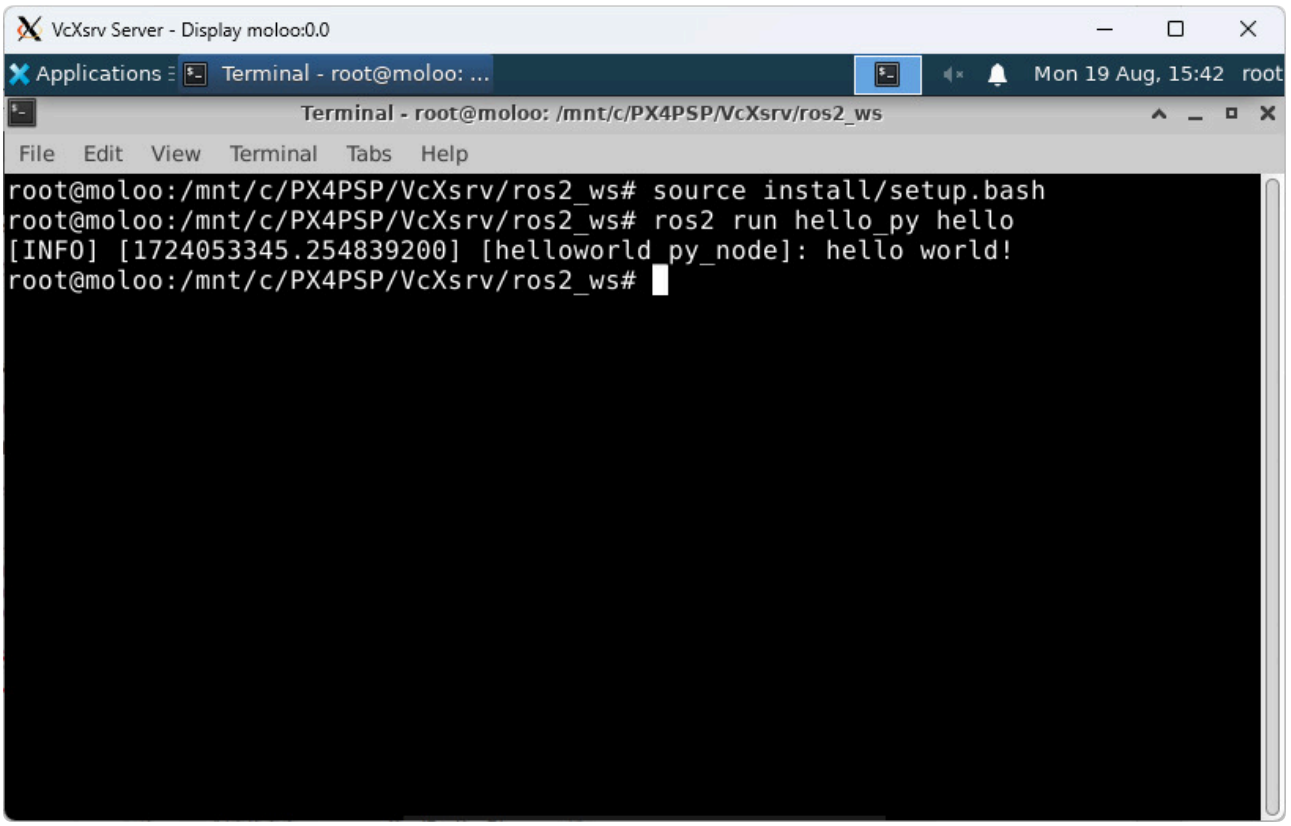
```
cd .. cd .. cd .. colcon build
```



Step5：

运行。

```
source install/setup.bash ros2 run hello_py hello
```



ROS2例程C++版（虚拟机，选做）

Step1：

创建ros2工作空间。

```
mkdir -p ros2_ws/src cd ros2_ws colcon build # 编译全部节点
```

Step2：

创建功能包。

```
cd src # 创建C/C++功能包 命令行 # 注意： # ament_cmake 使用 C/C++ 的功能包 # ament_python 使用 Python 功能包 ros2 pkg create node_name --build-type ament_cmake --dependencies rclcpp std_msgs
```

这里我们命名为hello，所以代码修改如下。

```
ros2 pkg create hello --build-type ament_cmake --dependencies rclcpp std_msgs
```

Step3：

创建节点源文件。

注：也可以在创建包的命令中，通过添加“--node-name node_name”选项，来自动创建节点，免去手动创建的过程。

```
# 进入 src 目录 cd hello/src/ touch hello.cpp
```

使用gedit编写代码，复制进如下的内容。

```
#include <chrono> #include "rclcpp/rclcpp.hpp" using namespace std::chrono_literals; class MyTimer : public rclcpp::Node { rclcpp::TimerBase::SharedPtr mTimer; public: MyTimer() : Node("MyTimer"){ auto timer_cb = this -> void { RCLCPP_INFO(this->get_logger(), "Hello !!!!"); }; this->mTimer = create_wall_timer(100ms, timer_cb); } }; int main(int argc, char * argv[]){ rclcpp::init(argc, argv); rclcpp::spin(std::make_shared<MyTimer>()); rclcpp::shutdown(); return 0; }
```

Step4：

编写 CMake 脚本。

```
cd .. # 1.打开 CMakeLists.txt gedit CMakeLists.txt # 2.增加节点编译脚本 add_executable(hello src/hello.cpp) ament_target_dependencies(hello rclcpp std_msgs) # 3.增加安装脚本 install(TARGETS hello DESTINATION lib/hello)
```

Step5：

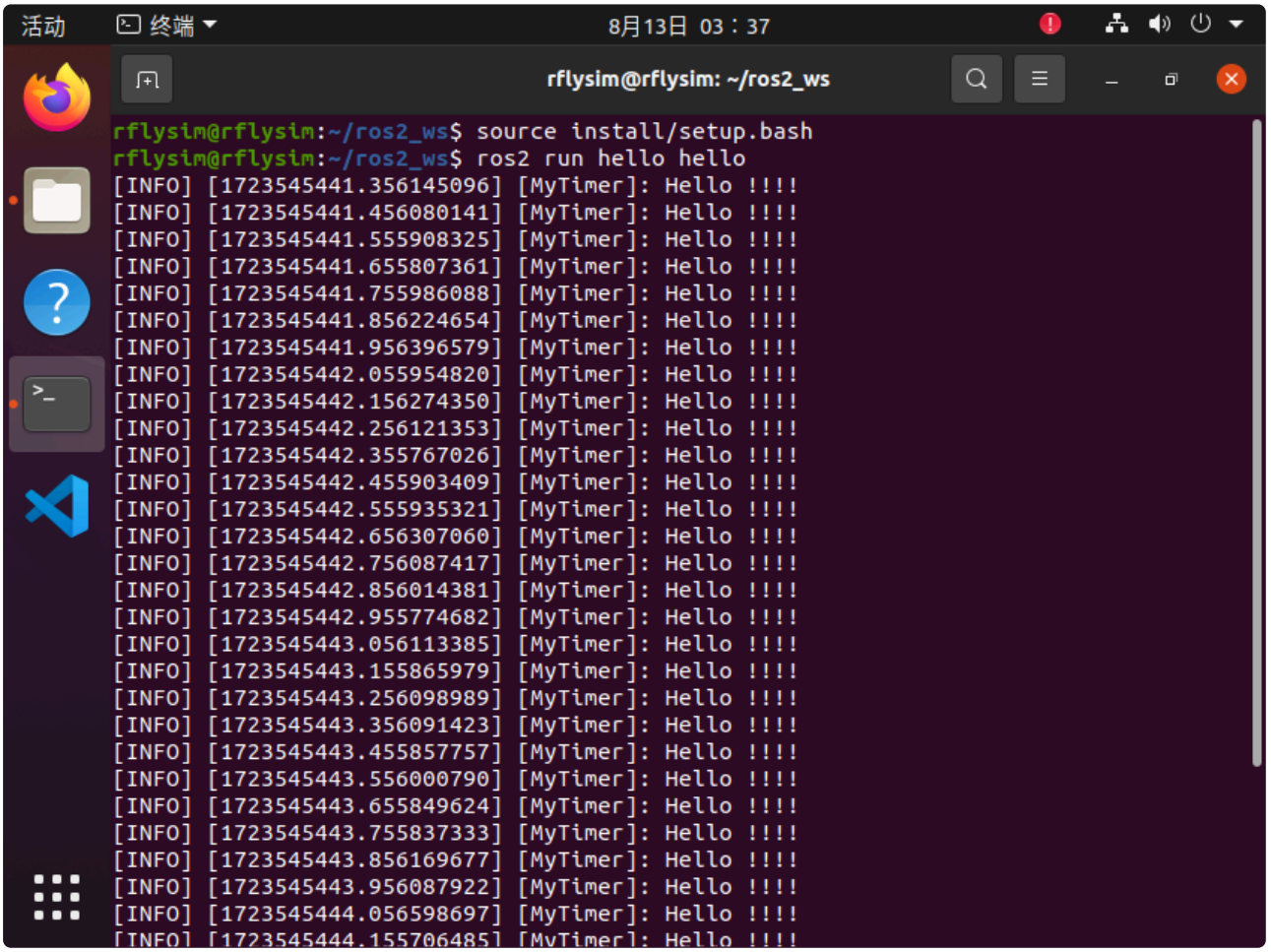
编译 ROS2 程序。

```
# 1.进入根目录 cd .. cd .. # 2.编译程序 colcon build --packages-select hello # 编译指定节点 colcon build # 编译全部节点
```

Step6：

运行 ROS2 程序。

```
# 1.加载环境 source install/setup.bash # 2.运行程序 ros2 run hello hello
```



ROS2例程python版（虚拟机，选做）

Step1：

如果按照上一节的内容，会创建好工作空间，无需继续创建。如果您还没有创建工作空间，请创建ros2工作空间。

```
mkdir -p ros2_ws/src cd ros2_ws colcon build # 编译全部节点
```

Step2：

进入工作空间，创建功能包和节点源文件。包名为hello_py，节点名为hello。

注：下面的命令，既可以创建功能包，还可以创建节点源文件，免去手动创建节点源文件的过程。C++的代码也可以通过添加“--node-name
node_name”选项，来创建节点。

```
cd src # 创建python功能包 命令行 # 注意： # ament_cmake 使用 C/C++ 的功能包 # ament_python 使用 Python 功能包 ros2 pkg create hello_py --build-type ament_python --dependencies rclpy --node-name hello
```

Step3：

节点源文件已经创建好，直接编辑源文件。

```
cd hello_py/hello_py # 进入该目录后，应该已经有hello.py文件，否则需要手动创建 gedit hello.py
```

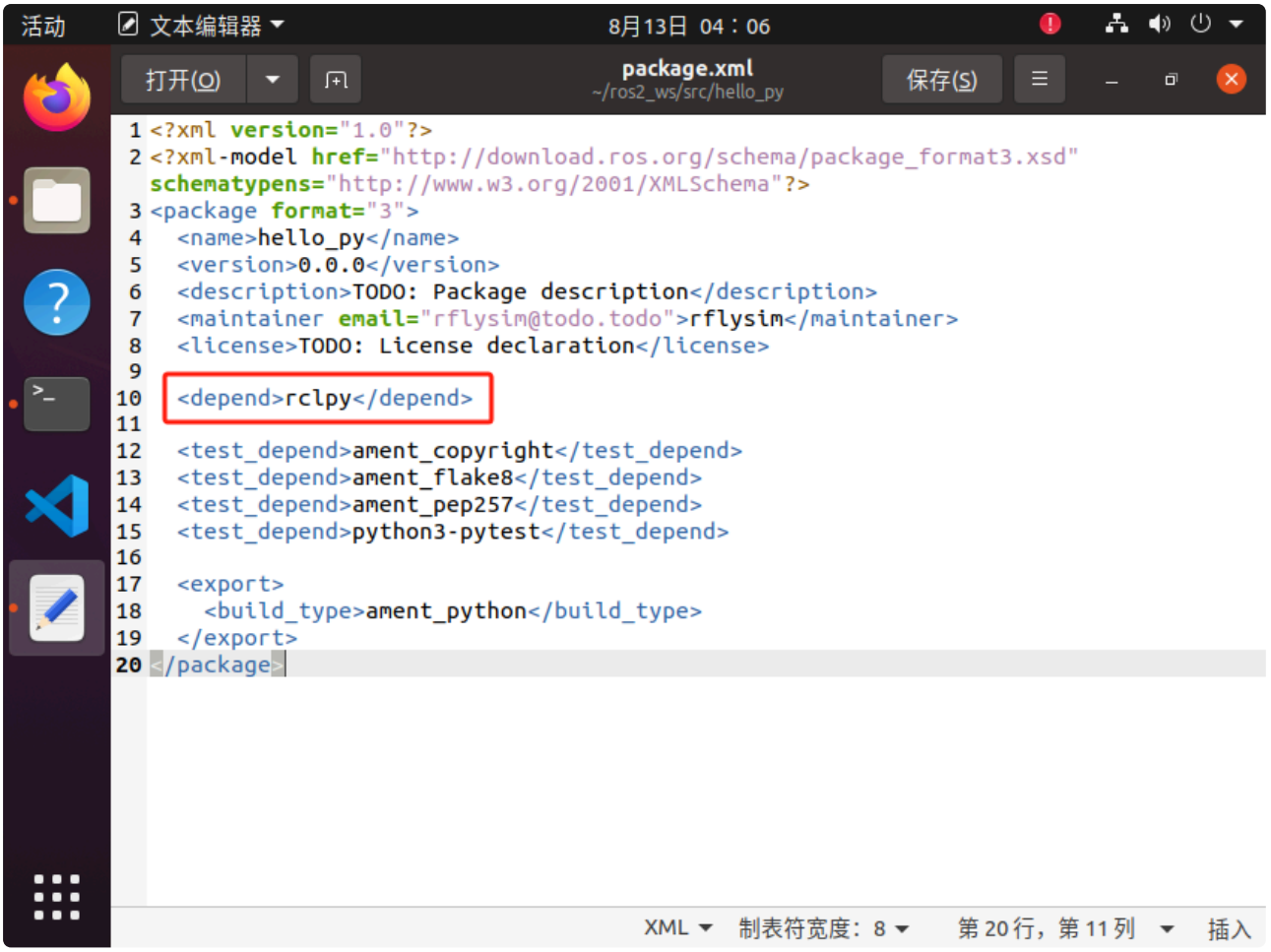
将下面代码复制进去，并保存。

```
import rclpy def main(): # 初始化 ROS2 rclpy.init() # 创建节点 node = rclpy.create_node("helloworld_py_node") # 输出文本 node.get_logger().info("hello world!") # 释放资源 rclpy.shutdown() if __name__ == 'main': main()
```

Step3：

编辑配置文件。与C++类似的，在步骤1创建功能包时所使用的指令也已经默认生成且配置了配置文件，不过实际应用中经常需要自己编辑配置文件，所以在此对相关内容做简单介绍，所使用的配置文件主要有两个， 分别是功能包下的package.xml与setup.py。

文件package.xml的内容如下，红色方框标记的内容可根据需要添加、删除和修改。



文件setup.py的内容如下，红色方框标记的内容可根据需要修改。

Step4：

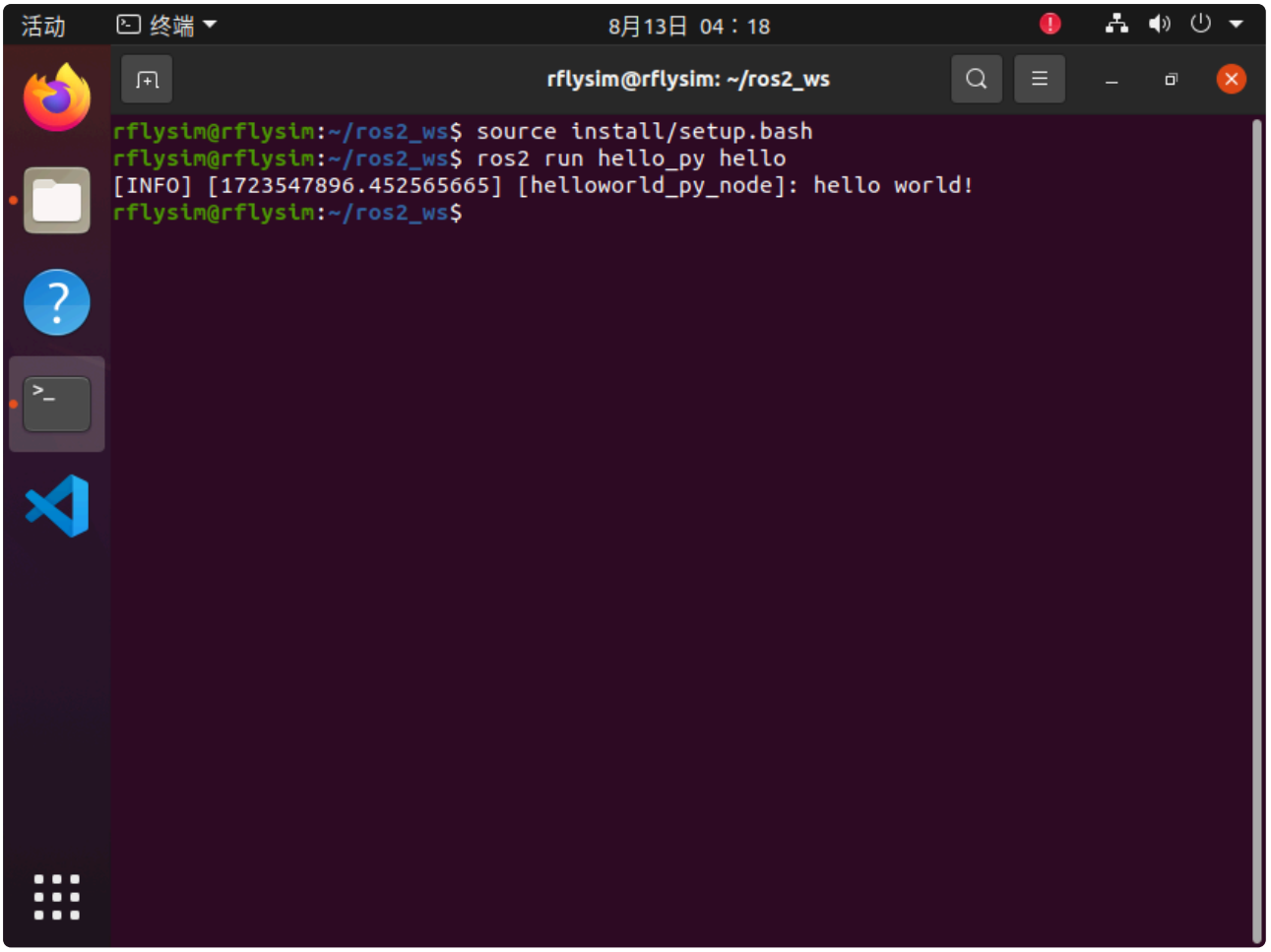
编译。

```
cd .. cd .. cd .. colcon build
```

Step5：

运行。

```
source install/setup.bash ros2 run hello_py hello
```



ROS2学习

ROS2官方网站：这是ROS2的官方发布平台，提供了ROS2的详细介绍、文档、教程以及最新的发布信息。可以通过此网站获取ROS2的官方安装指南、API文档、教程等资源。

官网链接（示例，请根据实际情况查找）：https://docs.ros.org/en/foxy/index.html

OS1（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ROS2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。