

ROS 人工智能基础

颜丙功 彭兆举 柳 建 尤芳怡 主编

清华大学出版社

北 京

内 容 简 介

本书面向 ROS 2 与人工智能融合的需求,通过具体、清晰的操作步骤讲解和实例分析,提供循序渐进的学习参考。本书共 8 章,覆盖 ROS 2 开发环境搭建、ROS 2 与相机及 ABB 机器人联调、ROS 2 与 YOLO 及 OpenVINO 融合等技术内容。通过本书学习,读者能够了解 ROS 2 与人工智能的融合方法,为复杂机器人项目的开发与优化奠定基础。

本书可作为高等院校机器人、自动化、计算机、人工智能等相关专业 ROS 课程的教材,也可作为机器人开发者、AI 工程师、自动化领域工程技术人员学习 ROS 的参考书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。举报:010-62782989, beiqinquan@tup.tsinghua.edu.cn。

图书在版编目(CIP)数据

ROS 人工智能基础 / 颜丙功等主编. -- 北京:清华大学出版社, 2026. 10. -- ISBN 978-7-302-71771-3

I . TP242

中国国家版本馆 CIP 数据核字第 20264DH730 号

责任编辑:王 定

封面设计:周晓亮

版式设计:思创景点

责任校对:成凤进

责任印制:丛怀宇

出版发行:清华大学出版社

网 址: <https://www.tup.com.cn>, <https://www.wqxuetang.com>

地 址:北京清华大学学研大厦A座 邮 编:100084

社 总 机:010-83470000 邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者:河北龙大印务有限公司

经 销:全国新华书店

开 本:185mm×260mm 印 张:14.75 字 数:332千字

版 次:2026年10月第1版 印 次:2026年10月第1次印刷

定 价:69.80元

产品编号:113550-01

前言 PREFACE

ROS(Robot Operating System) 作为当前应用广泛的开源机器人系统，在分布式架构、多线程处理、实时性、安全性等方面具有无可比拟的优势，已成为机器人研发的事实标准平台。ROS 活跃的开发社区和丰富的工具集与函数库，以及卓越的模块化设计与跨平台兼容性，为开发者高效构建复杂机器人及智能制造系统提供了坚实基础。

人工智能 (Artificial Intelligence, AI) 是引领全球科技发展的核心驱动力之一，正深刻重塑包括机器人技术在内的众多领域。将 AI 与 ROS 深度融合，能够显著提升机器人的智能化水平，增强其在复杂任务中的适应性与表现力，拓展机器人应用的广度和深度。然而，AI 与 ROS 融合过程中面临着多技术交叉、接口适配复杂、实操门槛高等问题，制约了相关应用的落地。

本书从理论到实操，覆盖 ROS 2 与 AI 融合应用的关键环节。首先从 ROS 2 开发环境的搭建入手，介绍相关软件及硬件的基础知识，然后围绕机器人控制、机器视觉、深度学习及模型部署等内容，系统拆解基础原理与操作流程，最后通过综合开发实战案例，将各模块内容融会贯通，本书注重实用性，案例贴合实际应用场景，力求为读者提供清晰、实用的学习参考，让读者在掌握基础操作的同时，具备解决实际问题的能力。

本书专为 ROS+AI 相关学习者与实践者打造，可作为高等院校机器人、自动化、计算机、人工智能等相关专业 ROS 课程的教材，也可作为从事机器人研发工作的工程技术人员、机器人技术爱好者，尤其是需要落地 ROS 2 与 AI 融合应用的开发者的实操手册。

本书的撰写离不开同仁的支持与帮助。我们非常感谢 ROS 全球开发者社区，感谢在编写过程中给予指导的各位专家，同时感谢团队成员，大家的通力协作保障了书稿的顺利付梓。

限于编者水平，书中难免存在疏漏与不足之处，恳请广大读者不吝赐教，提出宝贵的指正意见。我们将持续吸纳反馈、迭代完善，与行业同仁共同推动 ROS2 与 AI 融合技术的发展与进步。



本书配有教学课件、教学大纲、电子教案和习题参考答案，读者可扫描下列二维码获取。



教学课件



教学大纲



电子教案



习题参考答案

编者

2026 年 6 月

目录 CONTENTS

第 1 章

ROS 2 开发环境搭建.....001

- ① 1.1 ROS 2 概述.....002
 - 1.1.1 ROS 介绍.....002
 - 1.1.2 从 ROS 到 ROS 2.....003
- ① 1.2 Linux 概述.....007
 - 1.2.1 Linux.....007
 - 1.2.2 Ubuntu.....008
- ① 1.3 开发环境搭建.....008
 - 1.3.1 VMware 虚拟机.....009
 - 1.3.2 AIXBoard.....014
- ① 1.4 ROS 2 安装与配置.....016
- ① 1.5 ROS 2 基础操作.....017
 - 1.5.1 ROS 2 常用指令及环境变量说明.....017
 - 1.5.2 ROS 2 工作空间与功能包创建.....021
 - 1.5.3 ROS 2 的可视化与开发调试工具介绍.....023
- ① 1.6 本章小结.....024
- ① 1.7 习题.....024

第 2 章

ROS 2 机器人控制.....026

- ① 2.1 MoveIt 2 路径规划与可视化.....027
 - 2.1.1 MoveIt 2 安装.....027
 - 2.1.2 URDF 文件的构建与应用.....030
 - 2.1.3 RViz 2 可视化.....034
 - 2.1.4 MoveIt 2 路径规划.....042
- ① 2.2 ROS 2 与 ABB 机器人联调.....047
 - 2.2.1 RobotStudio 端配置.....047
 - 2.2.2 ROS 2 系统配置.....054



2.3	micro-ROS	057	2.3.5	发布 / 订阅话题控制机器人	074
2.3.1	micro-ROS 概述	057	2.4	本章小结	083
2.3.2	安装 micro-ROS Agent	059	2.5	习题	083
2.3.3	搭建开发环境	060			
2.3.4	发布 / 订阅话题控制 LED 灯 闪烁	063			

第 3 章

ROS 2 机器视觉 084

3.1	机器视觉概述	085	3.3.2	驱动程序的安装与调试	091
3.2	工业相机简介	086	3.3.3	相机网络配置	093
3.3	ROS 2 机器视觉开发	088	3.3.4	ROS 2 节点开发	101
3.3.1	ROS 2 与 Basler 相机接口及 驱动程序安装	088	3.4	本章小结	103
			3.5	习题	103

第 4 章

ROS 2+YOLO 深度学习 104

4.1	YOLO 概述	105	4.3.3	模型部署与应用	128
4.1.1	YOLO 算法结构	105	4.4	ROS 2 与 YOLO 的集成	128
4.1.2	YOLO 模型训练	106	4.4.1	ROS 2 目标检测节点设计	129
4.1.3	YOLO 模型推理	106	4.4.2	ROS 2 与 YOLO 集成实践	132
4.2	YOLO 深度学习环境搭建	107	4.4.3	ROS 2 目标检测优化 2.0	132
4.2.1	WSL 2 安装	107	4.5	实例：基于 YOLO 的目标检测 应用案例	139
4.2.2	Linux 分发版安装	109	4.5.1	应用场景描述	139
4.2.3	CUDA 安装	109	4.5.2	系统架构与实现	139
4.2.4	cuDNN 安装	112	4.5.3	案例演示与结果分析	142
4.2.5	Anaconda 安装	115	4.6	本章小结	143
4.2.6	PyCharm 安装	116	4.7	习题	143
4.3	YOLO 模型的训练	118			
4.3.1	数据集整理与标注	119			
4.3.2	YOLO 模型训练流程	121			



第 5 章

ROS 2+OpenVINO 模型部署 145

- ⑤ 5.1 OpenVINO 简介 146
 - 5.1.1 OpenVINO 概述 146
 - 5.1.2 OpenVINO 的发展历史 146
 - 5.1.3 OpenVINO 的核心优势 147
- ⑤ 5.2 OpenVINO 与 ROS2 集成 148
 - 5.2.1 OpenVINO 版本选择与下载 148
 - 5.2.2 OpenVINO 环境安装与配置 150
 - 5.2.3 ros2_opencv_toolkit 整体架构 151
 - 5.2.4 ros2_opencv_toolkit 的构建与安装 153
 - 5.2.5 ROS 2 与 OpenVINO 的实际应用 157
- ⑤ 5.3 OpenVINO 模型调用 161
- ⑤ 5.4 OpenVINO 加速模型 163
 - 5.4.1 添加 OpenVINO™ APT 存储库 164
 - 5.4.2 模型格式转换 165
 - 5.4.3 程序开发：ROS 2、OpenVINO 与 YOLOv8 集成 165
- ⑤ 5.5 本章小结 175
- ⑤ 5.6 习题 176

第 6 章

ROS 2 人工智能实践 1 177

- ⑤ 6.1 模型训练与部署 178
- ⑤ 6.2 ROS 2 算法推理 181
- ⑤ 6.3 Qt 人机交互界面设计 187
 - 6.3.1 ROS 2 与 Qt 集成原理 187
 - 6.3.2 ROS 2 与 Qt 的具体案例 188
- ⑤ 6.4 本章小结 201
- ⑤ 6.5 习题 201

第 7 章

ROS 2 人工智能实践 2 203

- ⑤ 7.1 外骨骼机器人概述 204
- ⑤ 7.2 ROS 2 外骨骼开发 204
- ⑤ 7.3 本章小节 211
- ⑤ 7.4 习题 211



第 8 章

ROS 2 人工智能实践 3.....213

- ④ 8.1 Go2 机器人概述..... 214
- ④ 8.2 ROS 2 驱动 Go2 仿真 214
- ④ 8.3 环境准备与仿真开发..... 214
- ④ 8.4 本章小结..... 225
- ④ 8.5 习题..... 225

参考文献

..... 226



第 1 章

ROS 2 开发环境搭建

本章将系统介绍 ROS 2 的基础知识、如何在 Ubuntu 系统中搭建 ROS 2 开发环境、常用的开发与调试工具，以及创建和管理 ROS 2 工作空间与功能包的过程，为后续开发实践奠定基础。

■ 本章教学目标

- (1) 了解 ROS 2 的架构及其与 ROS 的区别。
- (2) 掌握在 Ubuntu 系统中安装和配置 ROS 2 的方法。
- (3) 掌握基于虚拟机和 AIXBoard 的开发平台搭建方法。
- (4) 掌握 VSCode、Git 等工具的基础操作。
- (5) 掌握 ROS 2 工作空间与功能包的创建与管理方法。
- (6) 掌握 RViz 2、rqt 的基本使用方法。

■ 思维导图





1.1 ROS 2 概述

本节将系统介绍 ROS 与 ROS 2 的技术演进历程，分析 ROS 2 相较于 ROS 的改进之处，解析 ROS 2 的技术框架。

1.1.1 ROS 介绍

2007 年，斯坦福大学人工智能实验室提出了“共建共享”“避免重复造轮子”的机器人开发平台构想。2009 年，Morgan Quigley、吴恩达及 Willow Garage 机器人公司的工程师们在 IEEE 国际机器人与自动化会议配套的开源软件研讨会上发表了论文“ROS: An Open-Source Robot Operating System”，正式向外界公布了机器人操作系统 (ROS)。ROS 诞生后，由 Willow Garage 公司进一步加以完善，如今由 Open Robotics 负责维护与推广。历经多年迭代，ROS 已成为开源机器人开发的主要平台，切实解决了硬件兼容、算法复用及模块化等机器人开发的核心问题，成为机器人协同创新开发的稳固基础。ROS 的发展历程及各发行版本的相关信息如表 1-1 所示。

表 1-1 ROS 发行版本

发行版	发布日期	标识	停止维护日期
ROS Noetic Ninjemys	2020 年 5 月 23 日		2025 年 5 月
ROS Melodic Morenia	2018 年 5 月 23 日		2023 年 6 月 27 日
ROS Lunar Loggerhead	2017 年 5 月 23 日		2019 年 5 月
ROS Kinetic Kame	2016 年 5 月 23 日		2021 年 4 月
ROS Jade Turtle	2015 年 5 月 23 日		2017 年 5 月



(续表)

发行版	发布日期	标识	停止维护日期
ROS Indigo Igloo	2014 年 7 月 22 日		2019 年 4 月
ROS Hydro Medusa	2013 年 9 月 4 日		2015 年 5 月
ROS Groovy Galapagos	2012 年 12 月 31 日		2014 年 7 月
ROS Fuerte Turtle	2012 年 4 月 23 日		—
ROS Electric Emys	2011 年 8 月 30 日		—
ROS Diamondback	2011 年 3 月 2 日		—
ROS C Turtle	2010 年 8 月 2 日		—
ROS Box Turtle	2010 年 3 月 2 日		—

1.1.2 从 ROS 到 ROS 2

ROS 并非传统意义上用于进程管理与调度的操作系统，而是构建于其他操作系统之上的结构化通信层。它提供了标准化的通信机制，使不同的机器人硬件与软件模块能够便捷地实现交互与协作。此通信机制基于消息传递，可使各个模块异步运行与通信，提升了系



统的灵活性与可扩展性。

1. ROS 的问题

随着应用场景的持续拓展，ROS 暴露出一些问题。

(1) 实时性欠佳。ROS 依赖 TCPROS 通信协议，难以满足低延时、高实时性控制的要求。

(2) 安全性与稳定性存在问题。在多机器人协同工作时，ROS 的安全机制与容错能力不足。

(3) 跨平台支持受限。ROS 仅能在 Linux 平台运行，不支持 Windows 和 macOS 等系统，限制了其在多种硬件环境下的应用。

2. ROS 2 的改进之处

为解决上述问题，使 ROS 能更好地满足工业应用需求，ROS 2 应运而生。ROS 2 在以下几个方面进行了改进：

(1) 通信机制变革。ROS 2 将 DDS(Data Distribution Service) 作为底层通信中间件，借助 DDS 提供的质量服务策略，实现了实时性更佳、高效可靠的消息传输。

(2) 多平台支持与系统兼容性增强。ROS 2 实现了对多系统 (如 Linux、Windows 和 MacOS 等) 的跨平台支持，有助于扩大 ROS 2 的工业应用范围。

(3) 系统安全性与容错能力增强。ROS 2 通过引入安全策略与节点生命周期管理机制，有效提升了系统的安全性及容错能力。DDS 安全插件的运用，使 ROS 2 在数据传输过程中可通过加密与认证，降低了网络攻击或系统故障带来的风险。

(4) 模块化与可扩展性提高。ROS 2 在架构设计上更注重模块化与可扩展性，既保留了 ROS 中成熟的话题、服务等通信机制，又引入了改进的节点管理与资源调度策略，既保障了开发者对既有功能的持续使用，也为新功能的扩展预留了空间。

3. ROS 2 版本

自 2017 年首个 ROS 2 版本 Ardent Apalone 发布以来，ROS 2 不断迭代，日趋成熟，表 1-2 为 ROS 2 版本清单。

表 1-2 ROS 2 发行的版本

发行版	发布日期	标识	停止维护日期
Jazzy Jalisco	2024 年 5 月 23 日		2029 年 5 月
Iron Irwini	2023 年 5 月 23 日		2024 年 12 月 4 日



(续表)

发行版	发布日期	标识	停止维护日期
Humble Hawksbill	2022 年 5 月 23 日		2027 年 5 月
Galactic Geochelone	2021 年 5 月 23 日		2022 年 12 月 9 日
Foxy Fitzroy	2020 年 6 月 5 日		2023 年 6 月 20 日
Eloquent Elusor	2019 年 11 月 22 日		2020 年 11 月
Dashing Diademata	2019 年 5 月 31 日		2021 年 5 月
Crystal Clemmys	2018 年 12 月 14 日		2019 年 12 月
Bouncy Bolson	2018 年 7 月 2 日		2019 年 7 月
Ardent Apalone	2017 年 12 月 8 日		2018 年 12 月



(续表)

发行版	发布日期	标识	停止维护日期
beta3	2017 年 9 月 13 日	—	2017 年 12 月
beta2	2017 年 7 月 5 日	—	2017 年 9 月
beta1	2016 年 12 月 19 日	—	2017 年 7 月
alpha1 - alpha8	2015 年 8 月 31 日	—	2016 年 12 月

4. ROS 2 的核心组成部分

ROS 2 包含四个核心组成部分, 分别为通信 (plumbing)、工具 (tools)、功能 (capabilities) 和社区 (community), 具体内容如图 1-1 所示。



图 1-1 ROS 2 技术架构

(1) 通信。通信层作为 ROS 2 系统的核心部分, 肩负着各模块间数据交换的重要职责。ROS 2 选用 DDS 作为底层中间件, 实现了高效且低延迟的消息传输。通信层的设计秉持面向接口的编程思想, 借助消息接口将各个节点相互关联。节点间的依赖程度被降至最低限度, 实现了节点间的故障隔离及模块间的无缝整合, 这不仅让开发者能够聚焦于具体功能的实现, 而且有利于系统的维护与拓展, 保障了在多节点分布式系统中数据交互的稳定可靠, 如图 1-2 所示。

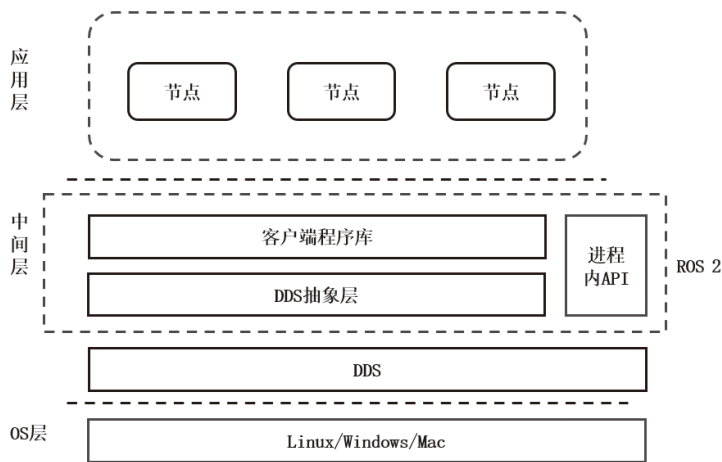


图 1-2 ROS 2 通信架构

(2) 工具。ROS 2 集成了多种开发工具, 旨在提升机器人应用开发的效率。启动管理 (Launch) 工具可助力开发者对多个节点进行统一调度, 简化系统部署流程, 保障系统的模



块化特征，为高效集成与拓展奠定基础；调试与监控工具能够为开发者提供实时数据、日志记录及故障诊断功能，对系统进行剖析，可快速定位并解决程序中的问题，提高开发效率；RViz 和 Gazebo 等可视化工具能够直观呈现机器人状态、传感器数据及环境参数，有助于开发者更为清晰地把握机器人行为，为系统分析与优化提供支撑；数据录制与回放工具可协助开发者记录并重现实验数据。

(3) 功能组件。ROS 2 具备丰富的功能组件，涵盖硬件驱动、传感器数据处理、算法等方面。基于标准接口，ROS 2 能够迅速实现多种传感器、执行器等的集成；依托丰富的算法库，ROS 2 可快速实现导航、运动规划和视觉处理等功能。此外，ROS 2 还具备参数管理和节点生命周期控制等功能，能够确保系统在复杂环境中高效且稳定地运行。ROS 2 丰富的功能组件使开发者能够在已有成熟模块的基础上，快速构建出满足个性化需求的机器人系统。

(4) 社区。ROS 2 社区汇聚了来自学术机构、企业及政府机构等的 ROS 2 开发者，通过开放性的交流与协作，共同推动了 ROS 2 的发展。社区成员积极贡献代码、分享案例、参与联合项目，促使 ROS 2 的功能持续完善。在社区的共同努力下，ROS 2 形成了一系列得到广泛认可的接口标准和开发规范，能够在不同平台和应用场景中实现灵活适配。社区还借助论坛、在线资源和研讨会等，为新手和专业人士提供了大量技术支持与学习机会。这种开放性协作的生态环境，使 ROS 2 得以快速迭代，为 ROS 2 的创新发展注入了持续动力。

1.2 Linux 概述

Linux 操作系统源于 1991 年林纳斯·托瓦兹在芬兰赫尔辛基大学开发的适用于 x86 架构的 Linux 内核。Linux 内核发布后，迅速吸引了全球开发者的关注，并通过 GNU 项目提供的工具、库及应用程序，演变成了完整的操作系统。自诞生以来，Linux 在服务器、超级计算机、嵌入式系统及移动设备等领域的应用不断拓展。发展至今，Linux 已经成为服务器的主流选择，也成为桌面系统和工业控制系统的重要成员。Linux 以其开源特性、稳定内核和灵活架构，成为 ROS 2 开发的标准平台。

1.2.1 Linux

Linux 内核采用 monolithic 架构，主要用 C 语言编写，辅以少量汇编代码处理与硬件相关的特定任务。模块化设计使得 Linux 内核可以在运行时动态加载或卸载驱动程序，从而具备较好的灵活性和可移植性。Linux 系统采用内核与用户软件分离的分层结构，保证了系统的安全性和稳定性。在 Linux 众多发行版中（如图 1-3 所示），有的专注于提供稳定高效的服务器环境，有的致力于为桌面用户提供直观友好的体验，还有的面向嵌入式设备和特定工业应用。这种多样化的发行版本，使用户可以根据自身需求进行选择。Ubuntu 是最受欢迎的 Linux 发行版，具有简洁直观的用户界面，易于安装和维护，使用广泛。

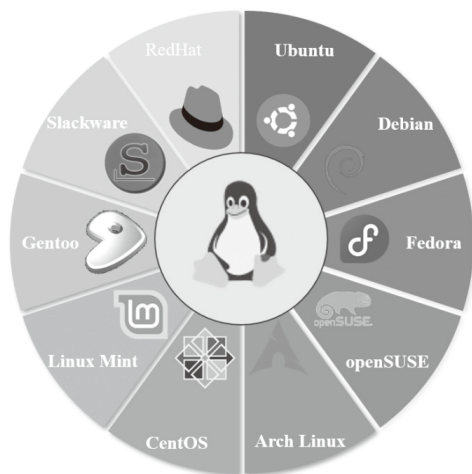


图 1-3 Linux 常用发行版

1.2.2 Ubuntu

Ubuntu 是基于 Debian 的 Linux 发行版，由英国企业 Canonical Ltd. 于 2004 年发布。Canonical 的创始人 Mark Shuttleworth 希望通过 Ubuntu 为广大用户提供简单、稳定且易于使用的操作系统。自 2004 年发布以来，Ubuntu 每六个月推出一个新版本，每两年发布一个长期支持版本 (LTS)。这种双轨发布模式既确保了系统能够及时引入最新的技术与功能，也可对稳定性要求较高的用户提供长达五年的支持周期。

Ubuntu 充分继承了 Debian 稳定的软件包管理体系，并在此基础上进行了大量优化。图形化的安装程序和系统更新工具简化了系统安装和维护过程，使普通用户可以轻松使用。对 Wayland 显示服务器的支持及对 Snappy 包管理系统的探索，进一步提升了系统的安全性和响应速度。Canonical 不仅通过技术支持促进 Ubuntu 的普及，同时鼓励用户和开发者积极参与社区建设，贡献代码、分享经验。Ubuntu 在持续更新的过程中积累了大量优质资源和成熟经验，为全球数以亿计的用户提供了坚实的技术支撑和持续创新的动力。

得益于 Ubuntu 长期支持版本的发布策略，开发者能够在稳定且不断更新的环境中构建和部署 ROS 2 应用。Ubuntu 完善的软件包管理系统和丰富的软件仓库使得依赖包的安装与管理非常简便，为 ROS 2 开发和测试提供了理想的开发环境。Ubuntu 的诸多优势，使其成为 ROS 2 的首选开发平台。

1.3 开发环境搭建

ROS 2 有两种开发环境搭建方案，即基于 VMware 虚拟机的搭建方案和基于 AIXBoard 的搭建方案。



1.3.1 VMware 虚拟机

VMware 是一款系统虚拟化软件,用户可借助其在单一物理主机上创建并运行多个操作系统,这不仅有助于提升硬件资源的利用效率,还能为软件开发营造安全且灵活的开发测试环境。VMware 于 1998 年由戴安·格林等人创立。该公司的首个产品 VMware Workstation 实现了桌面虚拟化,在开发者群体中迅速获得了广泛认可。随着市场需求的持续增长,VMware 相继推出适用于桌面、服务器和云平台的全系列虚拟化产品,如 VMware Player、VMware Fusion,以及面向企业级应用的 VMware vSphere 等。

基于 VMware® Workstation 17 Pro 安装 Ubuntu 22.04 系统的具体操作步骤如下。

(1) 下载并双击 VMware 虚拟机安装文件,单击“创建新的虚拟机”选项,在“新建虚拟机向导”对话框中,选取“自定义(高级)”模式,而后单击“下一步”按钮,如图 1-4 所示。

(2) 在“选择虚拟机硬件兼容性”对话框中,选取兼容至最高版本,单击“下一步”按钮,如图 1-5 所示。



图 1-4 VMware 安装向导



图 1-5 硬件兼容性

(3) 在“安装客户机操作系统”对话框中,选择“安装程序光盘映像文件(iso)”,再选择“浏览”,找到提前下载好的 Ubuntu 系统 iso 文件,单击“下一步”按钮,如图 1-6 所示。(Ubuntu.iso 文件下载方法:打开 <https://mirrors.ustc.edu.cn/>,找到 ubuntu-releases,选择 22.04,选择 ubuntu-22.04.5-desktop-amd64.iso,开始下载。)

(4) 输入安装信息,单击“下一步”按钮。

(5) 选择虚拟机安装位置,建议选择剩余容量在 60 GB 以上的硬盘分区。

(6) 处理器配置,建议根据实际情况进行选择,最大不可超过主机核数,最小不要小于 4 核,本书选择如图 1-7 所示,单击“下一步”按钮。



图 1-6 系统映像文件



图 1-7 处理器配置

(7) 设置虚拟内存，可以默认设置推荐内存 (如图 1-8 所示)，后续可根据需要进行调整，单击“下一步”按钮。

(8) 进入“网络类型”对话框，选择“使用网络地址转换 (NAT)” (如图 1-9 所示)，单击“下一步”按钮。



图 1-8 虚拟机内存



图 1-9 选择网络

(9) 如图 1-10 所示，在“选择 I/O 控制器类型”对话框中，选择推荐项 (LSI Logic)，单击“下一步”按钮。

(10) 如图 1-11 所示，在“选择磁盘类型”对话框中，选择推荐项 (SCSI)，单击“下一步”按钮。



图 1-10 I/O 控制器类型



图 1-11 磁盘类型

(11) 如图 1-12 所示，在“选择磁盘”对话框中，选择“创建新虚拟磁盘”（首次安装虚拟机必须创建新磁盘，后续若重新安装，可选择“使用现有虚拟磁盘”），单击“下一步”按钮。

(12) 如图 1-13 所示，在“指定磁盘容量”对话框中，填写最大磁盘容量，并选择“将虚拟磁盘拆分成多个文件”。本书设定为 120 GB 并选择将其进行拆分，单击“下一步”按钮。默认指定磁盘文件，继续单击“下一步”按钮。



图 1-12 选择磁盘



图 1-13 分配磁盘

(13) 操作完成后，可以看到如图 1-14 所示页面，已准备好创建虚拟机的配置。如果需要修改参数，可以单击“自定义硬件”按钮进行修改。最后单击“完成”按钮启动虚拟机，选择 Try or Install Ubuntu，进入 Ubuntu 安装步骤。



图 1-14 完成创建虚拟机

(14) 在安装界面左列的语言列表中，选择 Chinese，单击“安装 Ubuntu”按钮。在键盘布局界面中选择 Chinese(如图 1-15 所示)，单击 Continue 按钮。

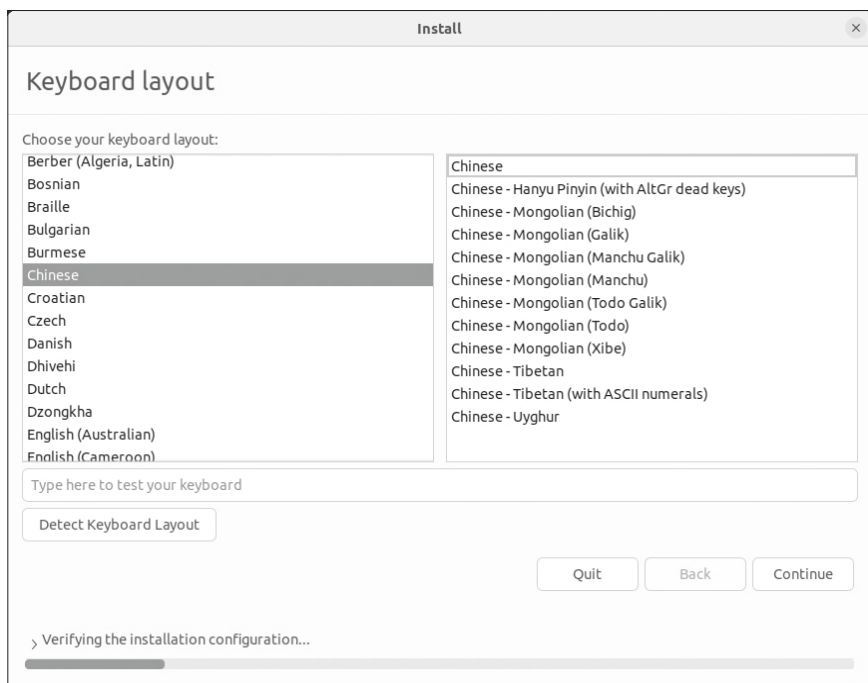


图 1-15 键盘语言设置

(15) 进入更新和其他软件对话框，选择 normal installation(normal installation 代表标准安装，包括浏览器、媒体浏览器等；minimal installation 代表最小安装，只安装浏览器等轻量化的必要工具)，如图 1-16 所示，单击 Continue 按钮。在安装类型对话框中，默认清除选项，单击“现在安装”按钮，会跳出分区确认窗口，单击 Continue 按钮。

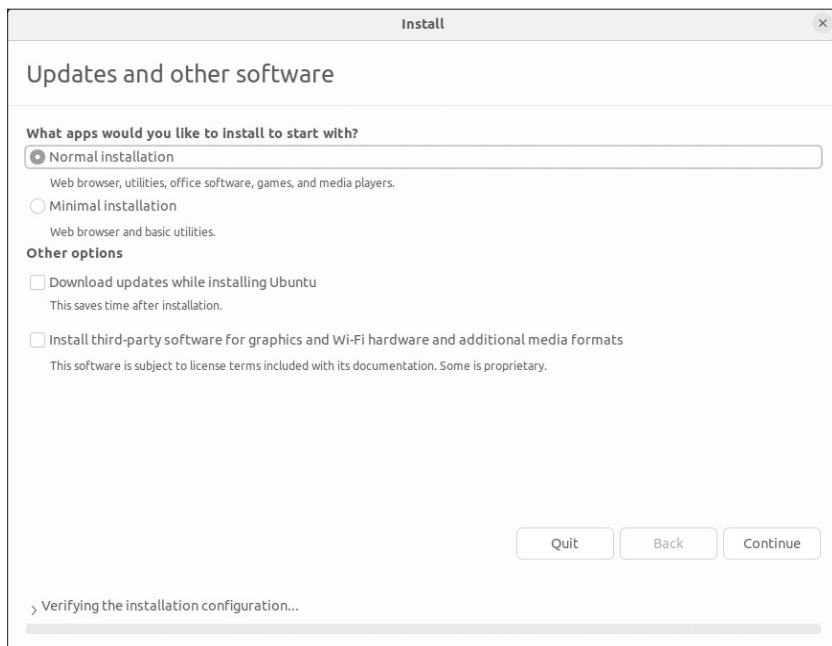


图 1-16 系统软件安装

(16) 在位置设置中选择所在位置，单击 **Continue** 按钮，进入用户设置对话框，设置用户名和密码 (如图 1-17 所示)，建议不要太复杂。单击 **Continue** 按钮，正式开始系统安装过程。

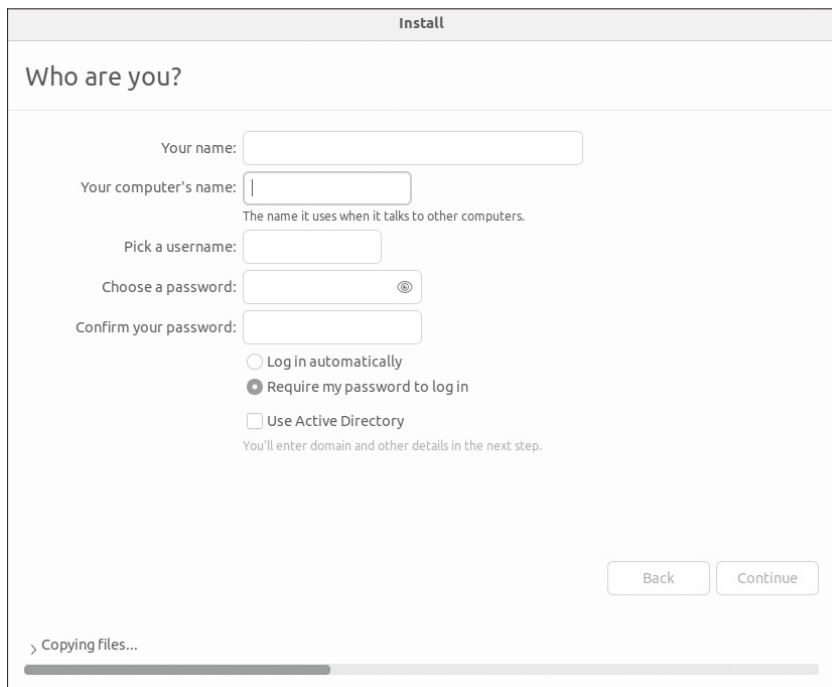


图 1-17 用户设置



(17) 安装完成后会跳出提示框，选择“现在重启”，在系统登录界面单击用户名，输入密码，进入创建的 Ubuntu 系统中。

1.3.2 AIxBoard

AIxBoard 开发板是一款入门级人工智能硬件平台 (如图 1-18 所示)。作为一款类似于树莓派的 x86 架构迷你主机，AIxBoard 不仅支持 Ubuntu，还能运行完整版的 Windows 操作系统。该开发板搭载了一颗英特尔四核处理器，最高频率可达 2.9 GHz，并配备了集成核显 (iGPU)，能够满足图形处理和视频输出等任务要求；内置 64 GB eMMC，支持 LPDDR4x 2933MHz 内存，提供 4 GB、6 GB 或 8 GB 多种容量配置，可适应不同应用对内存容量的需求；内置蓝牙和 Wi-Fi 模块，支持 USB 3.0、HDMI 视频输出、3.5 mm 音频接口及 1000Mbps 以太网接口，在网络连接和多媒体处理方面表现出色；接口兼容 Jetson Nano 及树莓派 GPIO，可最大程度复用现有的开发资源，能够使开发者轻松耦合各种传感器 (如图 1-19 所示)。作为边缘计算引擎，AIxBoard 既可用于人工智能产品的验证与开发，也可作为机器人控制板，为机器人研发提供高效、稳定的支持。AIxBoard 可运行 Visual Studio、OpenVINO、OpenCV 等主流开发工具，不仅为初学者提供了易用的边缘开发平台，也为专业开发人员提供了灵活的硬件环境，有力推动了人工智能和物联网领域的创新与应用。



图 1-18 AIxBoard 开发板

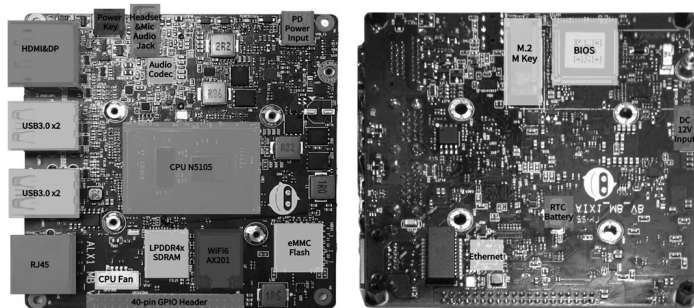


图 1-19 核心部件及接口

在 AIxBoard 中安装 Ubuntu 系统的具体操作步骤如下。



(1) 制作 U 盘启动盘。下载 USB Rufus 工具。Rufus 是一款格式化和创建 USB 启动盘的辅助工具。打开网址 <https://rufus.ie/zh/>，找到“其他版本 (GitHub)”，再找到 rufus-3.20p.exe，单击下载安装。

注：制作启动盘会将 U 盘格式化，U 盘中所有文件都将被清空，请注意存档。

(2) Rufus 软件安装完成后，插入 U 盘，启动 Rufus。如图 1-20 所示，在“引导类型选择”中，单击右侧“选择”按钮，找到 Ubuntu 系统映像文件并选中。

(3) “分区类型”和“目标系统类型”参考图 1-20，分别选择 MBR 与“BIOS 或 UEFI”即可。



图 1-20 Rufus 制作盘

(4) “文件系统”和“簇大小”保持默认选项 (Large FAT32 与 32 K 字节)，上述配置完成后，单击“开始”按钮。如果跳出“检测到 ISOHybrid 镜像”对话框，选择“以 ISO 镜像模式写入”，等待状态栏中进度条拉满后，Ubuntu 系统盘制作完成。

(5) 修改开机启动项顺序，改为从 USB 启动。将制作好的启动盘插入 AIXBoard 的 USB 接口中，启动 AIXBoard，连续按下键盘上的 F2 键，直至 Boot Menu 界面出现，使用键盘上的方向键将光标移到 Boot 并选择 KingstonDataTraveler 3.0PMAP, Partition 1，将其置于首位。

(6) 上述操作完成后，按下键盘上的 F10 键保存并退出。最后重新启动 AIXBoard，进入 Ubuntu 系统安装界面，后续步骤参考“1.3.1 VMware 虚拟机”。



1.4 ROS 2 安装与配置

本书使用的 Ubuntu 系统均为 22.04 版本，ROS 2 为 Humble Hawksbill 版本。安装过程中，需要从软件源服务器下载大量安装包文件，务必确保网络通畅。

1. 软件包更新

```
$ sudo apt update
[sudo] laser 的密码:
# 输入命令后提示输入密码，但密码将不显示，输入完成后直接按 Enter 键即可
```

2. 运行安装工具

启动新的终端，输入下列指令，等待片刻后，按照向导安装即可。

```
$ wget http://fishros.com/install -O fishros && bash fishros
```

输入 1，按下 Enter 键，一键安装 ROS，继续输入 1(更换系统源再继续安装) 后按下 Enter 键，接着继续输入 1(仅更换系统源) 并按下 Enter 键。

接着进入 ROS 2 相关安装选项。首先选择需要安装的 ROS 版本号，输入 1(humble 版本) 并按下 Enter 键，然后再次输入 1(桌面版) 并按下 Enter 键，等待安装工具自动下载安装即可。

注：若在安装过程中出现安装源错误问题，请选择更换安装源，可选择清华大学(以下简称清华) 或中国科学技术大学(以下简称中科大) 安装源。

3. 设置环境变量

启动新的终端，执行下列指令设置环境变量：

```
$ source /opt/ros/humble/setup.bash
```

4. 验证

启动一个新的终端，输入 ros2，验证是否安装成功。

```
$ ros2
```

若安装成功，则输出如图 1-21 所示信息。

在此可能得到未找到命令的错误提示，这是因为环境变量未成功设置，虽然执行了 source/opt/ros/humble/setup.bash，但每次重启新的终端，都需重新设置环境变量。为解决这一问题，请输入下列指令：

```
$ echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
```



```

laser@laser-virtual-machine: ~
laser@laser-virtual-machine:~$ ros2
usage: ros2 [-h] [--use-python-default-buffering]
          Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

Options:
  -h, --help            show this help message and exit
  --use-python-default-buffering
                        Do not force line buffering in stdout and instead use
                        the python default buffering, which might be affected
                        by PYTHONUNBUFFERED/-u and depends on whatever stdout
                        is interactive or not

Commands:
  action      Various action related sub-commands
  bag         Various rosbag related sub-commands
  component   Various component related sub-commands
  daemon      Various daemon related sub-commands
  doctor      Check ROS setup and other potential issues
  interface   Show information about ROS interfaces
  launch      Run a launch file
  lifecycle   Various lifecycle related sub-commands
  multicast   Various multicast related sub-commands

```

图 1-21 ROS 2 输出信息

1.5 ROS 2 基础操作

ROS 2 操作涵盖常用指令及环境变量设置、工作空间与功能包创建、开发调试等方面。

1.5.1 ROS 2 常用指令及环境变量说明

1. 终端启动

ROS 2 的操作均在 Terminal 终端进行。本书推荐终端为 Terminator。

\$ Ctrl+Alt+T	# 打开官方提供的终端
\$ sudo apt install terminator	# 安装 Terminator

运行上述指令之后，按照提示进行安装。安装完成后关闭终端，按下 Ctrl+Alt+T 键，即可打开 Terminator。

此终端的几个实用快捷键如下：

Ctrl+Alt+O	# 水平分割终端
Ctrl+Alt+E	# 垂直分割终端
Ctrl+Alt+C	# 复制
Ctrl+Alt+V	# 粘贴
Ctrl+Alt+F	# 搜索



2. VSCode

VSCode(Visual Studio Code)是由微软公司开发的免费、跨平台源代码编辑器。VSCode 基于 Electron 构建,可以在 Windows、macOS 和 Linux 等多种操作系统上稳定运行。VSCode 内置了丰富的功能,同时支持插件扩展,能满足各类编程语言和开发场景的需求,尤其在 ROS 2 开发群体中应用广泛。

启动 VMware 虚拟机,进入 Ubuntu 系统,打开浏览器,输入网址: <https://code.visualstudio.com/>,找到“.deb”点击下载。启动新的终端,定位到“下载”目录,输入解压指令进行解压安装。

```
$ cd ~/下载
# 在 Ubuntu 中, "Win+ 空格" 为中英文切换键
$ sudo dpkg -i ./code_1.77.0-1680085573_amd64.deb
# 输入 "/code" 后按下 Tab 键可实现自动填充,后续可用该方法判断目录下是否存在所需文件
```

验证是否安装成功:

```
$ code ./
# 启动 VSCode
# 若需使用 VSCode 对某一代码进行编辑,可以先进入其目录下,在终端输入 "code ./" 启动 VSCode
```

安装完成后,需要配置一些插件。

(1) 打开图 1-22 所示的扩展列表,在搜索栏中输入 Chinese,找到图 1-23 所示页面,下载安装简体中文插件。安装完成后,重启 VSCode,编辑器即变为中文版本。

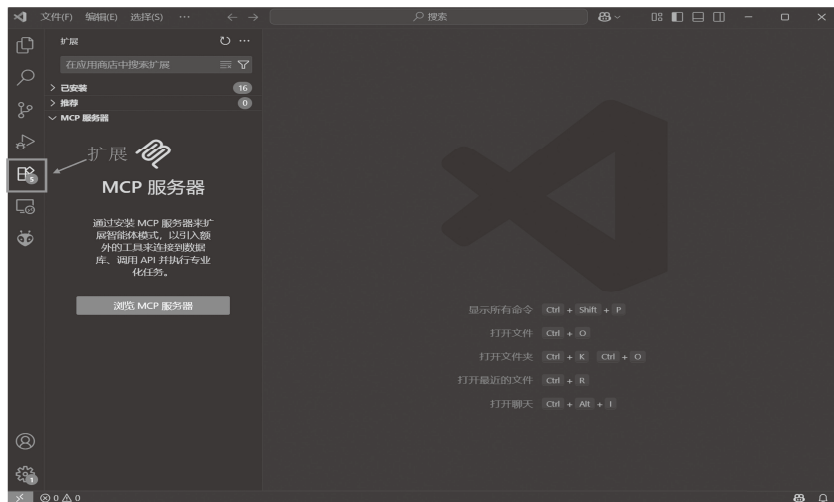


图 1-22 VSCode

(2) 安装 Python 提示插件。打开扩展列表,搜索 Python,如图 1-24 所示。安装完成后,使用 Python 时即可看到代码提示词。

(3) 安装 C++ 提示插件。搜索 C++ Extension Pack,如图 1-25 所示。安装完成后,使用 C++ 时即可看到代码提示词。



图 1-23 中文插件

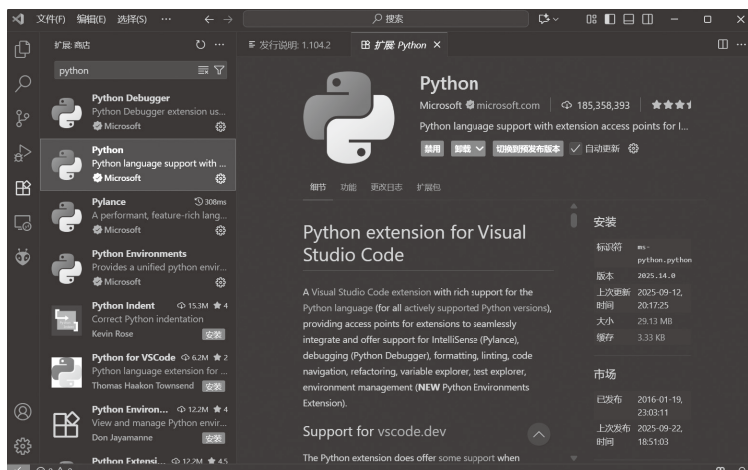


图 1-24 Python 插件

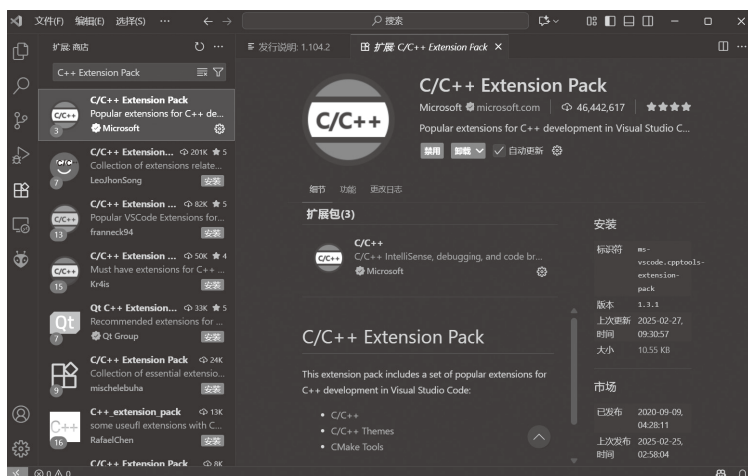


图 1-25 C++ 插件



3. Git 工具

Git 不仅能高效地处理大量并发指令，还能轻松管理分支、合并各类代码，确保项目代码的完整性与可追溯性。Git 在 ROS 2 的开发过程中发挥着重要作用。Git 工具下载十分简单，如下所示。

```
$ sudo apt install git      # 安装 Git 工具
$ git                       # 验证 Git 安装情况
```

4. ls 指令

ls 是 Linux 系统中的常用指令，用于查看目录所含内容，使用方法如下。

```
$ ls < 目录路径 >
```

5. cd 指令

cd 是 Linux 系统中的常用指令，用于打开工作目录。

```
$ cd                        # 若不指定 " 目录路径 "，则返回用户主目录
$ cd < 目录路径 >          # 返回目录内容
$ cd ..                     # 返回上一级目录
```

6. mkdir 指令

mkdir 指令用于创建文件夹，在 ROS 2 开发中常用于创建工作空间。

```
$ mkdir < 目录名 >          # 创建目录 / 文件夹
```

7. rm 指令

rm 指令用于删除目录中的一个或多个文件。

```
$ rm < 目录名 / 文件名 >
```

8. mv 指令

mv 指令用于移动目录，常用于将一个目录移动到另一个目录。

```
$ mv < 源目录 > < 目标目录 >
$ mv chapt1 chapt2          # 此例是将 chapt1 移动到 chapt2 目录下
```

9. cp 指令

cp 指令用于复制一个或多个文件。

```
$ cp < 源文件 > < 目标目录 >
$ cp chapt1 chapt2          # 将 chapt1 复制到 chapt2 目录下
```

10. sudo 指令

sudo 指令用于以系统管理员权限执行指令。



```
$ sudo [ 指令 ]  
$ sudo apt install ..... # 用于安装某软件包等  
$ sudo apt update # 用于更新
```

1.5.2 ROS 2 工作空间与功能包创建

在 ROS 2 开发中,工作空间和功能包的创建是系统开发的基础。正确配置和管理工作空间不仅有助于团队协作,同时也能提升代码的可维护性和扩展性。本节将详细介绍 ROS 2 工作空间的创建步骤。

1. ROS 2 工作空间概念

ROS 2 工作空间是一个用于管理 ROS 2 功能包的目录集合。ROS 2 工作空间是以 “***_ws” 命名的文件夹,该文件夹下包含以下子文件夹。

- (1) src: 存放功能包源代码,所有将要编译的功能包都需要置于该目录下。
- (2) install(编译后自动生成): 存放构建过程中生成的中间文件和最终安装的内容。
- (3) build(编译后自动生成): 记录构建过程的临时文件,可以在编译完成后删除,重新构建时自动生成。
- (4) log(编译后自动生成): 记录编译和运行时产生的日志信息,便于后续故障排查和性能分析。

2. 创建 ROS 2 工作空间

在正式开展开发工作前,首先需要初始化一个新的工作空间,可以按照以下步骤创建 ROS 2 工作空间:

```
$ mkdir -p ~/ros2_ws/src  
# -p 表示递归创建,创建完 ros2_ws 之后会在该目录下继续创建 src 文件夹  
$ cd ~/ros2_ws
```

ROS 2 工作空间创建成功后,进入工作空间根目录下,利用 colcon 工具对 src 目录下的功能包进行构建:

```
$ colcon build  
$ colcon build --packages-select < 功能包名 >  
# 在 ROS 2 中使用 colcon 编译时,若只需要编译特定的某个功能包,可以使用 --packages-select 指令  
# 运行 colcon build 时,若提示“未找到命令”,则说明没有正确安装 colcon,请执行下列指令,完成安装后对功能包进行构建  
$ sudo apt install python3-colcon-common-extensions
```

3. 创建 ROS 2 功能包

在工作空间搭建完毕后,即可创建 ROS 2 功能包。ROS 2 使用 ros2 pkg create 指令生成带有基础文件(如 CMakeLists.txt 和 package.xml)的功能包。功能包根据开发语言的不同,



可以分为 Python 和 C++ 两类。首先需要定位到 `src` 目录：

```
$ cd ~/ros2_ws/src # 在工作空间根目录下，进入 src 目录
```

如需创建名为 `my_python_package` 的基于 Python 的功能包，则采用 `ament_python` 作为构建类型，依赖 `rcipy`，创建指令如下。该指令会在 `src` 目录中自动生成 `setup.py`、`package.xml` 等模板文件。

```
$ ros2 pkg create <功能包名> --build-type ament_python --license Apache-2.0 --dependencies rcipy
```

如需创建名为 `my_cpp_package` 的基于 C++ 的功能包，则采用 `ament_cmake` 作为构建类型，并将依赖改为 `rcicpp`，创建指令如下。该指令会在 `~/ros2_ws/src` 目录下生成 `CMakeLists.txt`（描述了程序框架）、`package.xml`（描述了包的基本信息、版本、作者、许可证以及依赖关系）等模板文件。

```
$ ros2 pkg create <功能包名> --build-type ament_cmake --license Apache-2.0 --dependencies rcicpp
```

功能包创建指令中，`pkg` 为功能指令包含众多子指令；`create` 为 `pkg` 指令包中的“创建功能包”子指令；`--build-type` 表示所创建的功能包的构建类型，根据所使用的编程语言，可选 `ament_python` 或 `ament_cmake`；`--license` 用于声明功能包的开源协议，常用 `Apache-2.0`；`--dependencies` 表示依赖，根据所使用的编程语言，可选 `rcipy` 或 `rcicpp`。

```
$ ros2 pkg create <功能包名> --build-type <build-type> --license <license-type>
```

4. 功能包基础文件

功能包创建完成后，会生成一系列基础文件，这些文件构成了功能包的核心结构。其中，`package.xml` 为所有类型共有，`setup.py` 专属 `ament_python` 构建类型，`CMakeLists.txt` 专属 `ament_cmake` 构建类型，各文件的作用如表 1-3 所示。

表 1-3 功能包基础文件介绍

文件	主要作用
package.xml	ROS 2 包的“身份证”，定义包名、版本、描述、维护者、许可证、构建类型及依赖关系，确保包在工作空间中被正确识别、解析和安装。 主要修改：根据具体项目，添加所用依赖，如 <code>rcipy</code> 、 <code>std_msgs</code> 等， <code><depend>rcipy</depend></code> <code><depend>std_msgs</depend></code>
setup.py	Python 包安装脚本，基于 <code>setuptools</code> ，定义 Python 包的安装逻辑，确保包可被 <code>import</code> 并安装到 ROS 2 环境中。 主要修改： <code>entry_points['console_scripts']</code> 添加可执行节点，如 <code>['node_name = your_package_name.node_script:main']</code>
CMakeLists.txt	C++ 构建配置文件，使用 CMake 语法，用于声明最低 CMake 版本、项目名称、依赖查找、目标构建规则、安装规则以及包导出信息。其中 <code>ament_package()</code> 用于完成包注册与导出，通常置于文件末尾



1.5.3 ROS 2 的可视化与开发调试工具介绍

1. RViz 2

RViz 2(Robot Visualization 2) 是 ROS 2 的核心可视化工具，能够实时显示传感器数据、模型状态、环境地图等。用户在安装 ROS 2 时，RViz 2 会自动安装。启动 RViz 2 非常简便，只需在终端中输入以下指令：

```
$ ros2 run rviz2 rviz2
```

RViz 2 启动后，用户可以看到如图 1-26 所示界面。

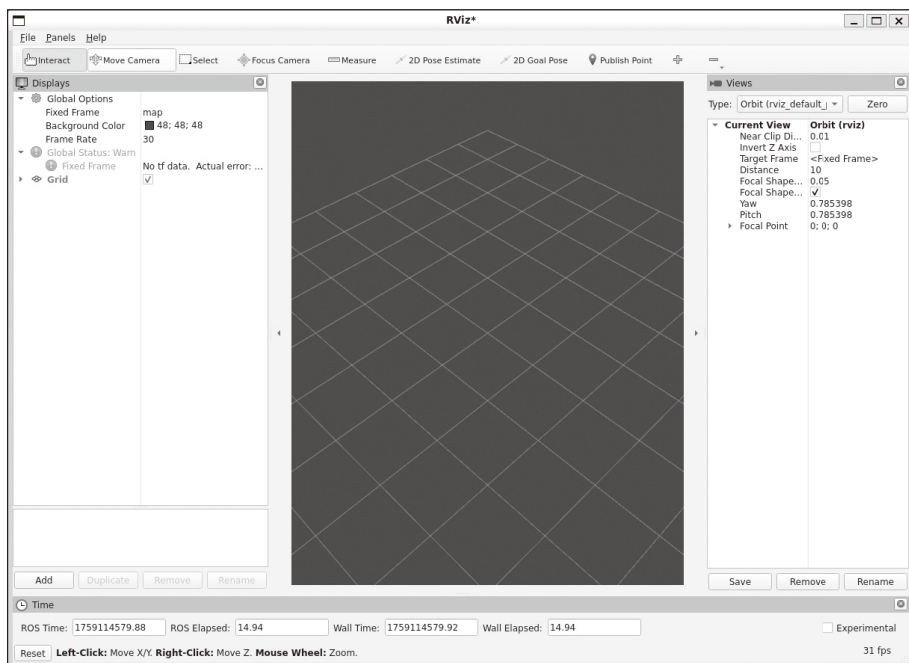


图 1-26 RViz 2 界面

- 工具栏：位于界面上方，提供视角控制、预估位姿设置、目标设置等功能按钮，还可以添加自定义插件。
- 显示列表：位于界面左侧，列出当前加载的所有显示项 (Displays)，用户可以在此区域添加、删除、复制、重命名插件，设置插件属性等。
- 三维视图：位于界面中央，是主要的显示区域，以三维方式显示数据。用户可以在此区域观察机器人模型、传感器数据、环境地图等。
- 视角设置：位于界面右侧，允许用户设置不同的观察角度，调整视图的显示方式。
- 时间显示区：位于界面下方，显示系统时间和 ROS 时间，帮助用户同步数据流。

RViz 2 通过插件提供丰富的功能，以满足不同的开发需求。

- (1) 显示插件：用于在三维视图中绘制不同类型的数据，如点云、图像、模型等。
- (2) TF 显示：可视化坐标变换 (TF) 信息，帮助开发者理解各坐标系之间的关系。



(3) 工具插件：提供与三维视图交互的工具，如选择、移动、旋转、测量等，方便用户进行操作。

(4) 视图控制：支持多种视图模式，如轨道视图、第一人称视图、俯视图等，满足不同的观察需求。

2. rqt

rqt 是 ROS 2 的图形化架构分析工具，基于 Qt 开发，既可以独立运行，也可以配合插件共同运行。通过下列指令可安装并启动 rqt (如图 1-27 所示)：

```
$ sudo apt install ros-<ros2_distro>-rqt # <ros2_distro> 为 ROS 2 版本，比如 humble  
$ rqt # 启动 rqt
```

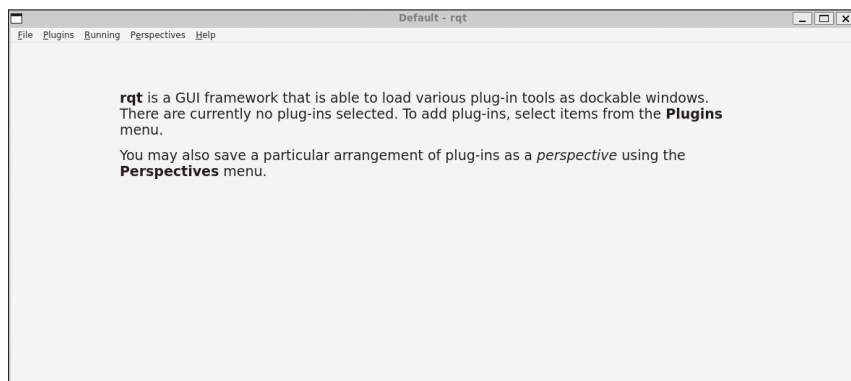


图 1-27 rqt 界面

1.6 本章小结

本章系统介绍了 ROS 2 开发环境的搭建流程，从 ROS 到 ROS 2 的关键改进，VSCode 和 Git 的安装、配置及其在 ROS 2 开发中的应用，以及 ROS 2 工作空间与功能包的创建流程，帮助开发者理解项目结构并掌握构建自定义 ROS 2 应用的基础操作。

1.7 习题

一、选择题

1. ROS 2 默认使用的通信中间件是 ()。
- | | |
|-----------|---------|
| A. TCPROS | B. DDS |
| C. HTTP | D. MQTT |



2. 以下 () 不是 ROS 2 相较于 ROS 的主要改进。

- A. 实时性提升
- B. 仅支持 Linux 系统
- C. 支持跨平台部署
- D. 增强安全机制

二、简答题

1. 简述 ROS 2 的四大组成部分及其功能。
2. 说明在 Ubuntu 中安装 ROS 2 的过程中需要注意的关键步骤。
3. 创建一个名为 my_ros2_ws 的工作空间，并在其中建立一个使用 Python 的示例功能包。
4. 如果在 Windows 系统中部署 ROS 2，你认为可能遇到哪些挑战？应如何解决？