# 格式化fmt：

**package：fmt**

**lib：fmt::fmt**

# 日志glog/logging.h：

**package：Glog**

**lib：glog::glog**

|  |  |
| --- | --- |
| 全局设置： | |
| google::InitGoogleLogging(argv[0]) | 初始化日志系统 |
| google::ShutdownGoogleLogging() | 关闭日志系统 |
| FLAGS\_logtostderr = true | 输出到stderr |
| FLAGS\_minloglevel = google::INFO | 日志最低级别 |

|  |  |
| --- | --- |
| LOG(INFO) / LOG(WARNING) / LOG(ERROR) / LOG(FATAL) | 初始化日志流 |

# 机器人ros/humble：

环境变量：

export DISPLAY='host.docker.internal:0'

source /opt/ros/humble/setup.bash

依赖配置：

package.xml

示例包：

turtlesim

局域网通信：

ROS\_DOMAIN\_ID (同一值)

|  |  |  |
| --- | --- | --- |
| ros2： | | |
| **运行** | ros2 run <pkg> <exe> | 运行可执行程序 |
| **包** | ros2 pkg executables <pkg> | 查看 |
| ros2 pkg create <pkg> --build-type <type> --dependencies <dep...> | 创建 |
| **结点** | ros2 node list | 列表 |
| ros2 node info <node> | 信息 |
| **话题** | ros2 topic list | 列表 |
| ros2 topic echo <topic> | 订阅 |
| ros2 topic pub <topic> <msgT> <value> | 发布 |
| **封装** | ros2 bag record <topic...> | 导出**数据** |
| ros2 bag play <path> | 复现 |
| **接口** | ros2 interface show <path> | 显示 |

|  |  |
| --- | --- |
| colcon： | |
| colcon build | **构建包** |
| source install/setup.bash | **读取包的环境变量** |

|  |  |
| --- | --- |
| ament\_cmake： | |
| ament\_package() | **处理包安装** |
| ament\_target\_dependencie(**<target> <dep...>**) | **链接ament依赖** |
| ament\_lint\_auto\_find\_test\_dependencies() | **查找并链接测试依赖** |

install(

TARGETS node

DESTINATION lib/${PROJECT\_NAME}

)

## 结点rclcpp/rclcpp.hpp：

|  |  |
| --- | --- |
| init(argc, argv) | 初始化 |
| spin(node\_ptr) | **进入结点的事件循环** |
| shutdown() | **清理程序** |
| RCLCPP\_INFO / RCLCPP\_WARN /  RCLCPP\_ERROR / RCLCPP\_FATAL(logger, fmt, arg...) | **输出日志** |
| **FutureReturnCode** spin\_until\_future\_complete(node\_ptr, future) | **等待异步结果** |

|  |  |  |
| --- | --- | --- |
| Rate(seconds) | | 循环频率**控制** |
| **方法** | sleep() | **阻塞** |

|  |  |  |
| --- | --- | --- |
| Node(name) | | 结点 |
| **属性** | get\_logger() | **日志系统** |
| **方法** | ::make\_shared(name) | **创建计数指针** |
| create\_wall\_timer(chrono\_duration, callback) | **创建定时执行器** |
| create\_publisher<msgT>(topic, qos) | **创建发布器** |
| create\_subscription<msgT>(topic, qos, callback) | **创建订阅器, callback(msgT::SharedPtr)** |
| create\_client<srvT>(service) | **创建客户端** |
| create\_service<srvT>(service, callback) | **创建服务端,**  **callback(srvT::Request::SharedPtr,**  **srvT::Response::SharedPtr)** |

|  |  |  |
| --- | --- | --- |
| Timerbase | | **定时执行器** |
| **属性** | ::SharedPtr | **计数指针** |
| time\_until\_trigger() | **触发时间** |
| **方法** | reset() | **重置** |
| cancel() | **取消** |

|  |  |  |
| --- | --- | --- |
| **消息：** | | |
| Publisher<msgT> | | 发布器 |
| **属性** | ::SharedPtr | **计数指针** |
| **方法** | publish(msg) | **发布** |
|  | | |
| Subscription<msgT> | | 订阅器 **(自动)** |
| **属性** | ::SharedPtr | **计数指针** |

|  |  |  |  |
| --- | --- | --- | --- |
| 服务： | | | |
| Client<srvT> | | | 客户端 |
| **属性** | ::SharedPtr | | **计数指针** |
| **方法** | async\_send\_request(request) | | **发送请求** |
| remove\_pending\_request(request\_id) | | **终止请求** |
|  | | | |
| Service<srvT> | | | 服务端 |
| **属性** | | ::SharedPtr | **计数指针** |
|  | | | |
| FutureAndRequestId<FutureT> | | | 异步响应 |
| **属性** | ::SharedPtr | | **计数指针** |
| request\_id | | 请求**编号** |
| **方法** | get() | | **等待**结果 (一次性) |

## 动作rclcpp\_action/rclcpp\_action.hpp：

|  |  |  |
| --- | --- | --- |
| ClientGoalHandle<actionT> | | 目标处理器 **(客户端)** |
| **属性** | **action\_msgs::msg::GoalStatus** get\_status() | 状态码 |

|  |  |  |  |
| --- | --- | --- | --- |
| Client<actionT> | | | 动作客户端 |
| create\_client<actionT>(node, name) | | | 创建动作客户端 |
| 属性 | ::WrappedResult | | 结果 **(Result指针)** |
| **属性** | **ResultCode** code | 状态码 |
| **actionT::Result** result | 结果 |
| ::SendGoalOptions | | 目标配置 |
| **属性** | goal\_response\_callback**(ClientGoalHandle::SharedPtr)** | **查看**目标筛选 |
| feedback\_callback**(ClientGoalHandle::SharedPtr,**  **actionT::Feedback::SharedPtr)** | **查看**目标反馈 |
| result\_callback**(::WrappedResult)** | **查看**目标结果 |
| 方法 | action\_server\_is\_ready() | | **服务器状态** |
| wait\_for\_action\_server(timeout = -1) | | **等待服务器** |
| async\_send\_goal(goal, option) | | **发送**目标 |
| async\_get\_result(goal\_handle, result\_callback = nullptr) | | 获取**异步结果** |
| async\_cancel\_goal(goal\_handle, cancel\_callback = nullptr) | | **取消**目标 |

|  |  |  |
| --- | --- | --- |
| ServerGoalHandle<actionT> | | 目标处理器 **(服务端)** |
| **属性** | get\_goal() | 目标 |
| is\_canceling() | 状态 |
| **方法** | canceled(result) | 取消**目标** |
| publish\_feedback(feedback) | **发布**反馈 |
| succeed(result) | **发布**结果 |

|  |  |  |
| --- | --- | --- |
| Server<actionT> | | 动作服务端 |
| create\_server<actionT>(node, name,  handle\_goal, handle\_cancel, handle\_accepted) | | 创建动作服务端 |
| 形参 | **GoalResponse** handle\_goal**(GoalUUID, actionT::Goal::SharedPtr)** | 目标筛选 |
| **CancelResponse** handle\_cancel**(std::shared\_ptr<ServerGoalHandle>)** | 取消请求 |
| handle\_accepted**(std::shared\_ptr<ServerGoalHandle>)** | 执行目标 |

## 接口rosidl\_interface\_packages：

**package：rosidl\_default\_generators**

**depend：**

**<build\_depend>rosidl\_default\_generators</build\_depend>**

**<exec\_depend>rosidl\_default\_runtime</exec\_depend>**

**<member\_of\_group>**rosidl\_interface\_packages**</member\_of\_group>**

|  |  |  |
| --- | --- | --- |
| \*.msg | \*.s | \*.action |
| string name  int64 id | int64 a  int64 b  ---  int64 sum | int32 g # \*::Goal  ---  int32[] r # \*::Result  ---  int32 f # \*::Feedback |

|  |  |
| --- | --- |
| ament\_cmake： | |
| rosidl\_generate\_interfaces(${PROJECT\_NAME} <\*.msg...> DEPENDENCIES) | 生成接口 |

## 消息\*\_msgs/msg：

**namespace：\*\_msgs::msg**

### 几何geometry\_msgs/msg：

### 标准std\_msgs/msg：

|  |  |  |
| --- | --- | --- |
| String | | 字符串 **(string.hpp)** |
| Int8 / Int16 / Int32 / Int64 | | 整型 **(int\*.hpp)** |
| Float32 / Float64 | | 浮点型 **(float\*.hpp)** |
| ColorRGBA | | 颜色 **(color\_rgba.hpp)** |
| **属性** | data | 数据 |
| ::SharedPtr | **计数指针** |

### 传感sensor\_msgs/msg：

## 服务\*\_srvs/srv：

**namespace：\*\_srvs::srv**

### 标准std\_srvs/srv：

|  |  |  |
| --- | --- | --- |
| Empty | | 空服务 **(empty.hpp)** |
| **属性** | ::Request | 请求 |
| ::Response | 响应 |