# 增强boost：

**namespace：boost**

## 格式format.hpp：

|  |  |  |
| --- | --- | --- |
| **format(fmt)** | | **初始化格式串** |
| **运算** | **% x** | **读入数据** |
| **方法** | **str()** | **转化为字符串** |

序列化serialization：

class MyClass : public **Base** {

int **mnId**;

int **mpArray[SIZE]**;

**friend class** boost::serialization::**access**; // serialization.hpp

**template<class Archive>**

**void serialize(Archive & ar, const unsigned int version)** {

ar & mnId;

ar & boost::serialization::**base\_object<Base>(\*this)**; // **base\_object.hpp**

ar & boost::serialization::**make\_array(mpArray, SIZE)**; // array.hpp

}

};

### 档案archive：

|  |  |  |
| --- | --- | --- |
| **序列化：** | | |
| **text\_oarchive(ostream)** | | **文本 (text\_oarchive.hpp)** |
| **binary\_oarchive(ostream)** | | **二进制 (binary\_oarchive.hpp)** |
| **运算** | **<< Obj** | **序列化** |

|  |  |  |
| --- | --- | --- |
| **反序列化：** | | |
| **text\_iarchive(istream)** | | **文本 (text\_iarchive.hpp)** |
| **binary\_iarchive(istream)** | | **二进制 (binary\_iarchive.hpp)** |
| **运算** | **>> Obj** | **反序列化** |

## 线程thread.hpp：

|  |  |
| --- | --- |
| **ref(obj)** | **实例化引用** |

|  |  |  |
| --- | --- | --- |
| **this\_thread** | **sleep\_for(chrono::duration)** | **当前线程休眠指定时长** |
| **sleep\_until(chrono::time\_point)** | **当前线程休眠到某时刻** |

|  |  |  |
| --- | --- | --- |
| **thread(func, ...)** | | **实例化线程** |
| **属性** | **joinable()** | **可阻塞** |
| **方法** | **join()** | **阻塞当前线程** |
| **detach()** | **后台执行，自动回收资源** |

|  |  |  |
| --- | --- | --- |
| **async(func, args...)** | | **异步执行，返回异步结果** |
| **unique\_future<T>** | | **实例化异步结果** |
| **方法** | **get()** | **等待结果 (一次性)** |

|  |  |  |
| --- | --- | --- |
| **mutex** | | **实例化互斥锁** |
| **属性** | **::scoped\_lock(mtx)** | **锁管理器，初始化时锁定，销毁时解锁** |
| **方法** | **lock / try\_lock()** | **锁定** |
| **unlock()** | **解锁** |

**原子atomic.hpp：**

|  |  |  |
| --- | --- | --- |
| **atomic<T>(value)** | | **原子操作类 (T有拷贝构造)** |
| **方法** | **load() / store(x)** | **读写** |

# 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 |
| LOG\_ASSERT(expr) | 断言 |

# 机器人ros：

Docker：[https://github.com/osrf/docker\_images](https://github.com/osrf/docker_images/tree/master/ros)

## humble：

环境变量：

source /opt/ros/humble/install/setup.bash # focal

source /opt/ros/humble/setup.bash # jammy

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 |
| ros2 launch <\*.py> |
| **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 package <pkg> | 列表 |
| ros2 interface show <path> | 定义 |

|  |  |
| --- | --- |
| colcon： | |
| colcon build | **编译工作空间** |
| source install/setup.bash | **加载环境变量** |

|  |  |
| --- | --- |
| ament\_cmake： | |
| ament\_target\_dependencie(**<target> <dep...>**) | **链接ament依赖** |
| **install(**  **TARGETS node**  **DESTINATION lib/${PROJECT\_NAME}**  **)** | **安装结点 (头文件，脚本类似)** |

|  |  |
| --- | --- |
| sudo： | |
| rqt | **系统监控** |
| rviz2 | **3D可视化** |
| ros2 run image\_view image\_view --ros-args -r image:=<topic> | **图像显示** |

### 通信rclcpp/rclcpp.hpp：

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

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

|  |  |  |
| --- | --- | --- |
| Parameter(name, value) | | 参数 |
| **方法** | get\_value / as\_bool / as\_bool\_array / as\_double / as\_double\_array /  as\_byte\_array / as\_int / as\_integer\_array / as\_string / as\_string\_array() | **获取值** |

|  |  |  |
| --- | --- | --- |
| Clock | | 时钟 |
| **方法** | **Time** now() | **当前时间 (buildin\_interfaces::msg::Time)** |

|  |  |  |
| --- | --- | --- |
| Node(name) | | 结点 |
| **属性** | get\_logger() | **日志系统** |
| get\_clock() | **时钟** |
| **参数** | declare\_parameter<T>(name, default) | **声明** |
| get\_parameter\_or(name, Parameter default) /  set\_parameter(Parameter) | **读写键对应值** |
| has\_parameter / undeclare\_parameter(name) | **查询 / 删除键** |
| **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：

#### 图像cv\_bridge/cv\_bridge.h：

编码：sensor\_msgs/image\_encodings.hpp

|  |  |
| --- | --- |
| CvImagePtr / CvImageConstPtr | **引用计数指针** |
| **CvImageConstPtr** toCvShare(msg, encoding) | **消息 → 指针** |

|  |  |  |
| --- | --- | --- |
| CvImage(header, encoding, image) | | **OpenCV图像** |
| **属性** | **std\_msgs::msg::Header header** | 时间戳 |
| **std::string encoding** | 格式 |
| **cv::Mat image** | 图像 |
| **方法** | **sensor\_msgs::msg::Image** **toImageMsg()** | **导出**消息 |

#### 几何geometry\_msgs/msg：

|  |  |  |
| --- | --- | --- |
| Quaternion | | 四元数 **(quaternion.hpp)** |
| **属性** | **float64 x / y / z / w** | 值 |

|  |  |  |
| --- | --- | --- |
| Vector3 | | 三元组 **(vector3.hpp)** |
| **属性** | **float64 x / y / z** | 值 |

|  |  |  |
| --- | --- | --- |
| Pose | | 位姿 **(pose.hpp)** |
| **属性** | **Vector3 position** | 位置 |
| **Quaternion orientation** | 朝向 |

|  |  |  |
| --- | --- | --- |
| Twist | | **速度 (twist.hpp)** |
| **属性** | **Vector3 linear** | 线性 |
| **Vector3 angular** | 旋转 |

|  |  |  |  |
| --- | --- | --- | --- |
| TransformStamped | | | 变换矩阵**时间戳 (transform\_stamped.hpp)** |
| **属性** | **std\_msgs::msg::Header header** | | 父坐标系 **(frame\_id)** |
| **std::string child\_frame\_id** | | 子坐标系 **(当前)** |
| Transformtransform | | **变换矩阵 (transform.hpp)** |
| **属性** | **Vector3 translation** | 平移 |
| **Quaternion rotation** | 旋转 |

#### 传感sensor\_msgs/msg：

|  |  |
| --- | --- |
| Image | 图像 **(image.hpp)** |
| CameraInfo | 相机 **(camera\_info.hpp)** |

#### 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** | **float32 r / g / b / a** | 值 |

|  |  |  |
| --- | --- | --- |
| Header | | 时间戳 **(header.hpp)** |
| **属性** | **buildin\_interfaces::msg::Time** stamp | 时间 |
| **string** frame\_id | 标识 |

### 服务\*\_srvs/srv：

**namespace：\*\_srvs::srv**

#### 标准std\_srvs/srv：

|  |  |  |
| --- | --- | --- |
| SetBool | | 空服务 **(set\_bool.hpp)** |
| **属性** | ::Request | 请求 |
| ::Response | 响应 |

### 欧式tf2：

|  |  |
| --- | --- |
| cmd： | |
| ros2 run tf2\_tools view\_frames | **绘制tf树** |
| ros2 run tf2\_ros tf2\_echo <src\_frame> <dst\_frame> | **输出变换** |

|  |  |
| --- | --- |
| **utils.h：** | |
| fromMsg(**geometry\_msgs::msg::Quaternion** in, tf2::**Quaternion** out) | 四元数转换 |

|  |  |  |
| --- | --- | --- |
| Quaternion | | 四元数 **(utils.h)** |
| 形参 | Quadword(x, y, z, w) | 元素 |
| setRotation(axis, angle) | 旋转 |
| 属性 | x / y / z / w() | 值 |
| 函数 | dot(q) | 点积 |
| normalize / normalized() | 单位化 |
| inverse() | 逆 |
| 方法 | ::getIdentity() | **创建**单位旋转四元数 |
| setEuler(yaw, pitch, roll) | **设置**欧拉角 |

|  |  |  |
| --- | --- | --- |
| Transform(rota, pos) | | 欧式变换 **(utils.h)** |
| 函数 | (vec3) | 点**变换** |
| inverse() | 逆 |

#### 转换tf2\_eigen：

|  |  |  |
| --- | --- | --- |
| **tf2\_eigen.hpp：** | | |
| **tf2** | **Eigen::Isometry3d** transformToEigen(t) | Transform /  TransformStamped |
| **geometry\_msgs::m**sg::TransformStampedeigenToTransform(t) | Isometry3d |
| **Eigen** | **geometry\_msgs::m**sg::Quaternion toMsg(q) | Eigen::Quaterniond |

#### 通信tf2\_ros：

|  |  |  |
| --- | --- | --- |
| Buffer(clock, cache\_time) | | 缓存 **(buffer.h)** |
| **方法** | lookupTransform(dst\_frame, src\_frame, time, timeout) | 查询变换 |

|  |  |
| --- | --- |
| **坐标监听器：** | |
| TransformListener(buffer) | 动态 /tf **(transform\_listener.h)** |

|  |  |  |
| --- | --- | --- |
| 坐标发布器： | | |
| StaticTransformBroadcaster<nodeT>(node, qos) | | 静态 /tf\_static **(static\_transform\_broadcaster.h)** |
| TransformBroadcaster<nodeT>(node, qos) | | 动态 /tf **(transform\_broadcaster.h)** |
| 方法 | sendTransform(**geometry\_msgs::m**sg::  TransformStamped t) | 发布 |

## noetic：

环境变量：

source /opt/ros/noetic/setup.bash # focal

示例包：

image\_view image\_view image:=<topic>

rviz rviz -d <\*.rviz>

|  |  |  |
| --- | --- | --- |
| ros： | | |
| **运行** | roscore | 启动ROS Master |
| rosrun <pkg> <exe> | 运行可执行程序 |
| roslaunch <\*.xml> |
| **结点** | rosnode list | 列表 |
| rosnode info <node> | 信息 |
| **话题** | rostopic list | 列表 |
| rostopic echo <topic> | 订阅 |
| rostopic pub <topic> <msgT> <value> | 发布 |
| **封装** | rosbag record <topic...> | 导出**数据** |
| rosbag play <path> | 复现 |
| **信息** | rosmsg package <pkg> | 列表 |
| rosmsg show <path> | 定义 |
| **服务** | rosservice list | 列表 |
| rosservice call <srv> <args...> | 调用 |
| **参数** | rosparam list | 键 |
| rosparam get <key> | **值**查询 |
| rosparam set <key> <value> | **值**设置 |
| rosparam dump <\*.yaml> | 保存 |
| rosparam load <\*.yaml> | 加载 |
| rosparam delete <key> | 移除 |

|  |  |
| --- | --- |
| catkin： | |
| catkin\_init\_workspace <./src> | **初始化**工作空间 |
| catkin\_create\_pkg <pkg> <dep...> | **创建**功能包 |
| catkin\_make install | **编译**工作空间 |
| source install/setup.bash | **加载环境变量** |

|  |  |
| --- | --- |
| catkin\_make： | |
| add\_dependencies(<target> <dep...>) | 链接依赖 |

### 通信ros/ros.h：

package：roscpp

|  |  |  |
| --- | --- | --- |
| 基本 | init(argc, argv, name) | 初始化 |
| shutdown() | **清理程序** |
| ok() | **系统状态** |
| ROS\_INFO / ROS\_WARN / ROS\_ERROR / ROS\_FATAL(fmt, arg...) | **输出日志** |
| ROS\_ASSERT(cond) | **断言** |
| 回调 | spin() | **进入事件循环** |
| spinOnce() | **处理一次回调** |
| **服务** | service::waitForService(srv) | **等待服务** |
| **参数** | param::get / param::set(key, value) | **读写键对应值** |
| param::has / param::del(key) | **查询 / 删除键** |

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

|  |  |  |
| --- | --- | --- |
| NodeHandle(ns="") | | 结点 |
| **方法** | advertise<msgT>(topic, qos) | **创建**发布器 |
| subscribe(topic, qos, callback) | **创建**订阅器 |
| serviceClient<srvT>(service) | **创建**客户端 |
| advertiseService(service, callback) | **创建**服务端 |

|  |  |  |
| --- | --- | --- |
| **消息：** | | |
| Publisher | | 发布器 |
| **方法** | publish(msg) | **发布** |
|  | | |
| Subscriber | | 订阅器 **(自动)** |

|  |  |  |
| --- | --- | --- |
| 服务： | | |
| ServiceClient | | 客户端 |
| **方法** | call(srv) | **发送请求** |
|  | | |
| ServiceServer | | 服务端 |

### 接口message\_generation：

CATKIN\_DEPENDS message\_runtime

**depend：**

<build\_depend>message\_generation</build\_depend>

<exec\_depend>message\_runtime</exec\_depend>

|  |  |
| --- | --- |
| catkin\_make： | |
| add\_message\_files / add\_service\_files / add\_action\_files(FILES <file...>) | 添加接口文件 |
| generate\_messages(DEPENDENCIES <dep...>) | 生成消息 |

# 配置yaml-cpp/yaml.h：

软件包：libyaml-cpp-dev

namespace：YAML

|  |  |
| --- | --- |
| Load(stream) | 从IO流加载 |
| LoadFile(file) | 从文件加载 |
| Dump(node) | 转存到字符串 |
| key\_to\_string(anything) | 转存到字符串 |

|  |  |  |
| --- | --- | --- |
| Node(value=NULL) | | 结点 |
| **属性** | IsNull / IsScalar / IsSequence / IsMap() | **空 / 标量 / 序列 / 映射** |
| **Node** operator[](key) | 键值**查找** |
| size() | **长度** |
| **begin / end()** | **指定位置迭代器** |
| **方法** | **as<T>()** | 标量**类型转换** |
| **push\_back(value\_or\_node)** | **新增序列值** |
| **bool remove(key\_or\_node)** | **删除键或结点** |