



Programmation robotique avec ROS2 en C++

Dr. Eng. Chiheb Ameer ABID

 /in/chiheb-ameur-abid
 chiheb.abid@gmail.com

 Janvier 2026
Chiheb Ameer ABID



Partie 1
└ Introduction

2 / 69

Plan

- 1 Introduction
 - Présentation de ROS (Robot Operating System)
 - Point sur les dernières distributions de ROS et ROS2
 - Tour d'horizon des robots compatibles
 - Installation et configuration
- 2 Architectures
- 3 Outils de visualisation et de simulation
- 4 Interface CLI de ROS2

Plan

1 Introduction

- Présentation de ROS (Robot Operating System)
- Point sur les dernières distributions de ROS et ROS2
- Tour d'horizon des robots compatibles
- Installation et configuration

2 Architectures

- Architecture de ROS2
- Architecture d'une application ROS2
- L'organisation des fichiers
- Présentation packages disponibles

3 Outils de visualisation et de simulation

4 Interface CLI de ROS2

Présentation

Motivation : problèmes de développement des applications robotiques

↳ Avant ROS

- ✘ Peu de réutilisation du code
- ✘ Réécrire sans cesse les pilotes de périphériques, l'accès aux interfaces du robot, la gestion des processus embarqués, les protocoles de communication inter-processus, etc.
- ✘ Recoder sans cesse les algorithmes standards
- ✘ Nouveau robot dans le laboratoire (ou dans l'usine) : recommencer à coder (en grande partie) à partir de zéro

Présentation

Historique

- Développé à l'origine en 2007 au laboratoire d'intelligence artificielle de Stanford
- Le développement s'est poursuivi chez l'entreprise Willow Garage fondée par Larry Page
 - ☞ Larry Page est le fondateur de Google
 - ☞ Willow Garage a construit un robot très sophistiqué le PR-2 (équipé de deux bras, d'une base mobile et avec des moyens de calcul importants)
- Willow Garage s'est fermée en 2013, et l'Open Source Robotics Foundation (OSRF) financé par plusieurs sociétés
 - ☞ Toyota a fait un don de 50 millions de dollars pour la fondation

Présentation

C'est quoi ?

- ROS (Robot Operating System) est un *système d'exploitation open-source*
 - ☞ Meta-système d'exploitation!?
 - ☞ Un ensemble de bibliothèques logicielles et d'outils qui vous aident à construire des applications robotiques qui fonctionnent sur une grande variété de plates-formes robotiques
- Les principales fonctionnalités de ROS sont les suivantes :
 - ☞ Un middleware de communication qui permet aux différents composants d'un robot de communiquer entre eux
 - ☞ Un système de gestion de packages qui permet de gérer les bibliothèques, les outils et les autres ressources nécessaires au développement d'applications ROS
 - ☞ Un ensemble d'outils de développement, tels que des IDE, des outils de débogage et des outils de visualisation, qui facilitent le développement d'applications robotiques

Présentation

ROS vs OS

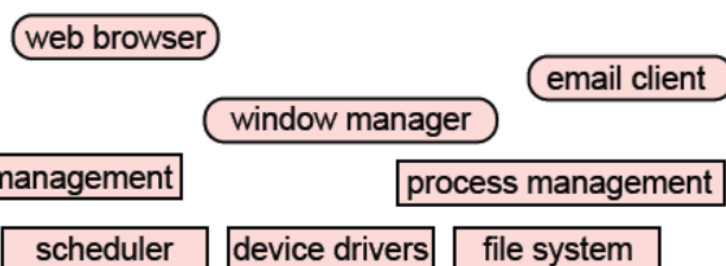
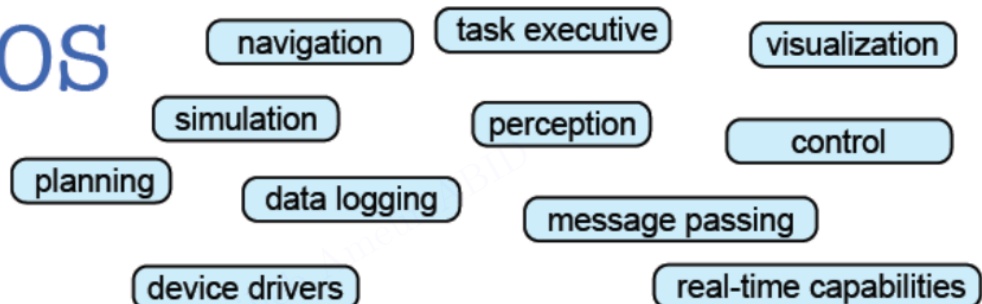
- ROS est un framework middleware robotique
- ROS possède deux **faces**
 - ① Le côté **services de type OS**
 - ☞ Abstraction matérielle
 - ☞ Abstraction matérielle (hardware abstraction)
 - ☞ Gestion des périphériques et drivers de bas niveau
 - ☞ Outils et bibliothèques pour des fonctionnalités courantes
 - ☞ Communication inter-processus (publication/souscription, services, actions)
 - ☞ Gestion des paramètres, logs, enregistrement de données (rosviz), visualisation (rviz), etc.
 - ② Le côté **écosystème de paquets robotiques**
 - ☞ Une très large collection de paquets open-source
 - ☞ Implémentations prêtes à l'emploi pour des algorithmes robotiques avancés : la cartographie simultanée et la localisation (SLAM), la planification, la perception, la vision, la manipulation, etc.

ROS2

Présentation



ROS



OS



ROS2

Présentation

Philosophie

➤ La philosophie de ROS se résume dans les 5 grands principes suivants :

① Peer to Peer

- ✚ Les systèmes ROS se composent de nombreux petits programmes (noeuds) qui se connectent les uns aux autres et échangent continuellement des messages

② Basé sur des outils (microkernel)

- ✚ Il existe de nombreux petits programmes génériques qui effectuent des tâches telles que la visualisation, la journalisation, le traçage de flux de données, etc.

③ Multi langages

- ✚ Actuellement, des bibliothèques clientes existent pour C++, Python, LISP, Java, JavaScript, MATLAB, Ruby, et bien plus encore

④ Léger

- ✚ Les conventions ROS encouragent les contributeurs à créer des bibliothèques/packages autonomes

⑤ Gratuit et open source

ROS2

Présentation

ROS

➤ La clef de ROS est la mise en place de 4 grands types de mécanismes permettant de construire une application robotique

- ✚ Plomberie, c'est à dire la connexion des composants logiciels entre eux quelque soit la répartition des noeuds de calcul sur un réseau,
- ✚ Outils, c'est à dire un ensemble de logiciels permettant d'analyser, d'afficher et de débbuger une application répartie,
- ✚ Capacités, c'est à dire des bibliothèques qui implémentent les fonctionnalités telles que la planification de tâches (SMACH), de mouvements (OMPL), la construction d'un modèle de l'environnement (Octomap),
- ✚ l'Ecosystème, c'est à dire un nombre suffisant d'utilisateurs tels que ceux-ci ont plus intérêt à collaborer plutôt qu'à reconstruire les mêmes outils.

ROS2

Plan

1 Introduction

- Présentation de ROS (Robot Operating System)
- Point sur les dernières distributions de ROS et ROS2
- Tour d'horizon des robots compatibles
- Installation et configuration

2 Architectures

- Architecture de ROS2
- Architecture d'une application ROS2
- L'organisation des fichiers
- Présentation packages disponibles

3 Outils de visualisation et de simulation

4 Interface CLI de ROS2

Point sur les dernières distributions de ROS

Point sur les dernières distributions de ROS

- Les versions de ROS/ROS2 sont publiées tous les ans
 - 🔗 Les versions majeures sont publiées tous les deux ans
- Melodic Morenia, publiée en mai 2018
 - 🔗 Supportée jusqu'au mai 2023
 - 🔗 Améliorations dans la stabilité, la performance, modularité et la documentation
- Noetic Ninjemys, publiée en mai 2020 (dernière version de ROS1)
 - 🔗 Supportée jusqu'au mai 2025
 - 🔗 Des améliorations dans la compatibilité avec Python 3
 - 🔗 Prise en charge de Ubuntu 20.04
 - 🔗 Migration vers Qt 5 et la mise à jour de nombreux packages

Point sur les dernières distributions de ROS2

Point sur les dernières distributions de ROS2

➤ Foxy Fitzroy, publiée en mai 2020

- ☞ La première version à long terme (LTS) de ROS2
- ☞ Intégration avec d'autres frameworks comme TensorFlow, Gazebo ou MoveIt

➤ Humble Hawksbill est une version majeure de ROS2, sortie le 24 mai 2022

- ☞ Supportée jusqu'à 2027
- ☞ Prise en charge de la nouvelle version du système d'exploitation Ubuntu 22.04
- ☞ Amélioration de la prise en charge des appareils embarqués, notamment l'ajout de support pour les architectures ARM64.

➤ Iron Irwini, publiée en 2 mai 2023

- ☞ Génération automatique de documentation API pour les packages Python
- ☞ Introduit l'introspection des services qui permet aux développeurs de récupérer des informations sur les services ROS2

➤ Jazzy Jalisco, publiée le 23 mai 2024

- ☞ Version LTS actuelle, supportée jusqu'en mai 2029
- ☞ Support natif d'Ubuntu 24.04 (Noble Numbat) + Windows 10/11
- ☞ Améliorations majeures sur la sécurité, la performance RMW (DDS), et l'intégration Zenoh (middleware alternatif)

ROS2

Plan

1 Introduction

- Présentation de ROS (Robot Operating System)
- Point sur les dernières distributions de ROS et ROS2
- Tour d'horizon des robots compatibles
- Installation et configuration

2 Architectures

- Architecture de ROS2
- Architecture d'une application ROS2
- L'organisation des fichiers
- Présentation packages disponibles

3 Outils de visualisation et de simulation

4 Interface CLI de ROS2

ROS2

Robots compatibles ROS

Robots compatibles ROS

➤ Liste de robots compatibles avec ROS/ROS2 : <https://robots.ros.org/>

📁 > 110 robots



[Fraunhofer IPA Care-O-bot](#)



[Videre Erratic](#)



[TurtleBot](#)



[Aldebaran Nao](#)



[Lego NXT](#)



[Shadow Hand](#)



[Willow Garage PR2](#)



[iRobot Roomba](#)



[Robotnik Guardian](#)



[Merlin miabotPro](#)



[AscTec Quadrotor](#)



[CoroWare Corobot](#)



[Clearpath Robotics Husky](#)



[Clearpath Robotics Kingfisher](#)



[Festo Didactic Robotino](#)

ROS2

Plan

1 Introduction

- Présentation de ROS (Robot Operating System)
- Point sur les dernières distributions de ROS et ROS2
- Tour d'horizon des robots compatibles
- Installation et configuration

2 Architectures

- Architecture de ROS2
- Architecture d'une application ROS2
- L'organisation des fichiers
- Présentation packages disponibles

3 Outils de visualisation et de simulation

4 Interface CLI de ROS2

ROS2

Installation

Installation de ROS2 (1/2)

- Lien pour l'installation de ROS2 sur une distribution debian
<https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html>
- Ajouter la clé du dépôt

```
1 sudo apt update && sudo apt install curl -y
2 sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o \
  /usr/share/keyrings/ros-archive-keyring.gpg
```

- Ajouter le dépôt dans la liste des dépôts

```
echo "deb [arch=$(dpkg --print-architecture) \
signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] \
http://packages.ros.org/ros2/ubuntu \
$(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee \
/etc/apt/sources.list.d/ros2.list > /dev/null
```

ROS2

Installation

Installation de ROS2 (2/2)

- Installer ROS2 Desktop : ROS, RViz, demos, tutorials.

```
sudo apt install ros-humble-desktop
```

- Installer les outils de développement

```
sudo apt install ros-dev-tools
```

- Ajouter la configuration de l'environnement dans le fichier `.bashrc`

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

ROS2

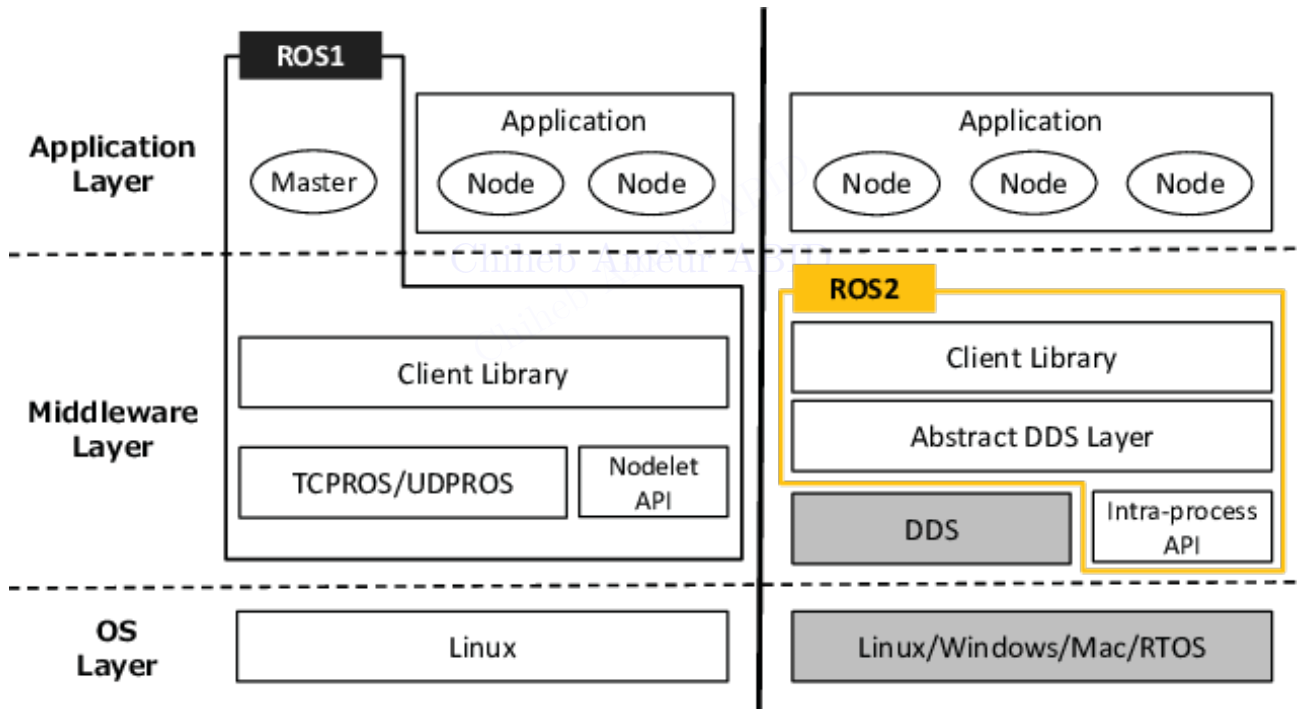
Plan

- 1 Introduction
- 2 Architectures
 - Architecture de ROS2
 - Architecture d'une application ROS2
 - L'organisation des fichiers
 - Présentation packages disponibles
- 3 Outils de visualisation et de simulation
- 4 Interface CLI de ROS2

Plan

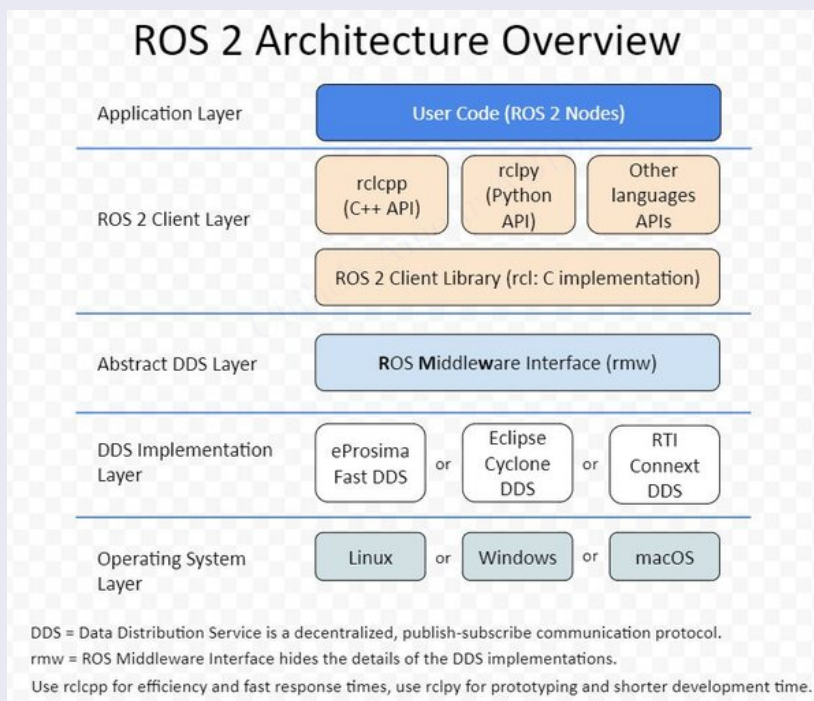
- 1 Introduction
 - Présentation de ROS (Robot Operating System)
 - Point sur les dernières distributions de ROS et ROS2
 - Tour d'horizon des robots compatibles
 - Installation et configuration
- 2 Architectures
 - Architecture de ROS2
 - Architecture d'une application ROS2
 - L'organisation des fichiers
 - Présentation packages disponibles
- 3 Outils de visualisation et de simulation
- 4 Interface CLI de ROS2

Architecture de ROS1 vs ROS2



Architecture de ROS2

- Par défaut, l'implémentation utilisée de DDS dans ROS2 est FastRTPS
- FastRTPS est une implémentation open-source de DDS



Architecture de ROS2

Data Distribution Service (DDS)

- DDS est une norme de middleware de communication de données temps réel
 - ✎ Défini par l'Object Management Group (OMG) et est utilisé par de nombreux systèmes critiques, tels que les systèmes embarqués et les systèmes de contrôle industriel
 - ✎ DDS est centré sur les données : distribuer des données entre des composants d'un système de manière fiable et efficiente.
- Les principaux services fournis par DDS sont les suivants :
 - ✎ La publication de données : permet à un composant de publier des données dans un espace de nommage
 - ✎ L'abonnement aux données : permet à un composant de s'abonner à des données publiées dans un espace de nommage
 - ✎ La livraison des données : DDS assure la livraison des données aux abonnés
 - ✎ La gestion de la qualité de service : DDS permet aux composants de spécifier les exigences de qualité de service pour la livraison des données
- DDS peut être déployé sur différents types de réseaux : TCP/IP, bluetooth, CAN, DDS Over-Serial (liaison série), etc.

ROS2

Plan

- 1 Introduction
 - Présentation de ROS (Robot Operating System)
 - Point sur les dernières distributions de ROS et ROS2
 - Tour d'horizon des robots compatibles
 - Installation et configuration
- 2 Architectures
 - Architecture de ROS2
 - Architecture d'une application ROS2
 - L'organisation des fichiers
 - Présentation packages disponibles
- 3 Outils de visualisation et de simulation
- 4 Interface CLI de ROS2

ROS2

Les différents éléments d'une application ROS2 (1/2)

Éléments d'une application ROS2

- L'architecture d'une application ROS2 est basée sur un modèle de communication par message
 - 🔗 **Nodes** : Un node est un exécutable qui utilise ROS2 pour communiquer avec d'autres nodes
 - 🔗 **Messages** : types de données ROS2 utilisés pour souscrire ou publier sur un topic.
- **Topics** sont des flux de données qui sont publiés par des noeuds et auxquels des noeuds peuvent s'abonner. Ils sont utilisés pour des tâches qui nécessitent une communication unidirectionnelle
- **Services** sont des interfaces de programmation d'applications (API) qui permettent aux noeuds de communiquer entre eux. Ils sont utilisés pour des tâches qui nécessitent une communication bidirectionnelle, telle que la demande d'un service à un autre noeud
- **Actions** permettent à un noeud de demander à un autre noeud d'exécuter une tâche de longue durée et d'obtenir un résultat

ROS2

Les différents éléments d'une application ROS2 (2/2)

Éléments d'une application ROS2

- **Paramètres** sont des valeurs stockées dans un serveur de paramètres qui peuvent être lues ou modifiées par les noeuds. Un paramètre peut être utilisé pour configurer le comportement d'un noeud ou pour partager des informations entre les noeuds
- **Bags** sont des fichiers qui enregistrent les données publiées par des noeuds
- **Composant** est une variante d'un noeud qui peut être chargé et déchargé dynamiquement dans un processus conteneur. Un composant permet de réduire l'empreinte mémoire et le temps de démarrage d'une application

ROS2

Les différents éléments

Node

↳ Définition

- ✎ Le processus associé à l'exécution d'un fichier exécutable dans un paquet ROS
- ✎ Les noeuds ROS utilisent une librairie client pour communiquer avec les autres noeuds. Les noeuds peuvent publier ou souscrire à des topics.
- ✎ Les noeuds peuvent fournir ou utiliser un service
- ✎ Les librairies client sont `roscpp` pour python et `roscpp` pour C++.

Plan

1 Introduction

- Présentation de ROS (Robot Operating System)
- Point sur les dernières distributions de ROS et ROS2
- Tour d'horizon des robots compatibles
- Installation et configuration

2 Architectures

- Architecture de ROS2
- Architecture d'une application ROS2
- L'organisation des fichiers
- Présentation packages disponibles

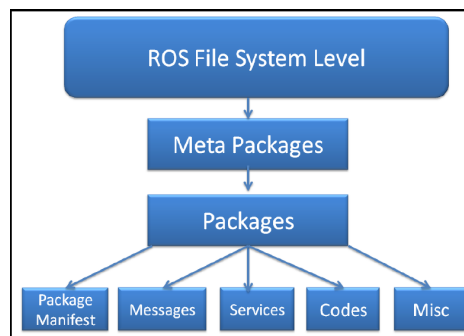
3 Outils de visualisation et de simulation

4 Interface CLI de ROS2

L'organisation des fichiers

Système de fichier de ROS

- ROS est plus qu'un framework de développement
 - ☞ C'est un méta système d'exploitation
- Il offre non seulement des outils et des bibliothèques, mais même des fonctions de type OS :
 - ☞ Abstraction matérielle
 - ☞ Gestion des packages
 - ☞ Chaîne d'outils de développement
- De manière similaire aux OS, les fichiers ROS sont organisés sur le disque dur d'une manière particulière



ROS2

L'organisation des fichiers

Système de fichier de ROS

- Packages
 - ☞ Unité de base de code ROS
 - ☞ Il s'agit d'un répertoire (projet)
 - ☞ Ils contiennent un ou plusieurs programmes ROS (noeuds), bibliothèques, fichiers de configuration, etc., qui sont organisés ensemble en une seule unité.
- Package manifest
 - ☞ Fichier au format XML contenu dans un package : `package.xml`
 - ☞ Contient des informations sur le package : l'auteur, la licence, les dépendances, les drapeaux de compilation, etc.

ROS2

L'organisation des fichiers

Système de fichier de ROS

↳ Metapackages

- ✚ Un ou plusieurs paquets apparentés qui peuvent être regroupés
- ✚ Ils sont des packages virtuels qui ne contiennent aucun code source ou fichier typique que l'on trouve habituellement dans les packages.

↳ Metapackages manifest

- ✚ Un ou plusieurs paquets apparentés qui peuvent être regroupés
- ✚ sont des packages virtuels qui ne contiennent aucun code source ou fichier typique que l'on trouve habituellement dans les packages.

↳ Messages (.msg)

- ✚ Les messages ROS sont un type d'informations envoyées d'un processus ROS à un autre
- ✚ Il est possible de définir un message personnalisé dans le dossier msg à l'intérieur d'un package (`mypackage/msg/MyMessageType.msg`).
- ✚ L'extension du fichier de message est `.msg`

L'organisation des fichiers

Système de fichier de ROS

↳ Services (.srv)

- ✚ Un service ROS est une sorte d'interaction requête/réponse entre les processus
- ✚ Les types de données de réponse et de demande peuvent être définis dans le dossier `srv` à l'intérieur du package `mypackage/srv/MyServiceType.srv`

↳ Dépôts (Repositories)

- ✚ La plupart des packages ROS 2 sont maintenus avec Git (souvent sur GitHub / GitLab)
- ✚ Un dépôt peut contenir un ou plusieurs packages ROS 2

L'organisation des fichiers

Exemple de structure d'un package

↳ Les fichiers et les dossiers constituant un package

```
ros_pkg
├── action
│   └── demo.action
├── CMakeLists.txt
├── include
│   └── ros_pkg
│       └── demo.h
├── msg
│   └── message.msg
├── src
│   └── demo.cpp
└── srv
    └── service.srv
```

Plan

1 Introduction

- Présentation de ROS (Robot Operating System)
- Point sur les dernières distributions de ROS et ROS2
- Tour d'horizon des robots compatibles
- Installation et configuration

2 Architectures

- Architecture de ROS2
- Architecture d'une application ROS2
- L'organisation des fichiers
- Présentation packages disponibles

3 Outils de visualisation et de simulation

4 Interface CLI de ROS2

Catégories de packages

- ROS2 fournit un ensemble de packages qui fournissent des fonctionnalités de base pour le développement de robots
- Ces packages peuvent être divisés en plusieurs catégories, telles que :
 - ✚ La communication : fournit des fonctionnalités pour la communication entre les noeuds, telles que la publication et l'abonnement à des messages, la communication bidirectionnelle via des services et la diffusion de données via des topics.
 - ✚ Les capteurs : fournit des fonctionnalités pour la communication avec les capteurs
 - ✚ Les actionneurs : fournit des fonctionnalités pour la commande des actionneurs
 - ✚ La navigation : fournit des fonctionnalités pour la navigation des robots, telles que la localisation, la cartographie et la planification des mouvements.
 - ✚ La perception : fournit des fonctionnalités pour la perception des robots, telles que la détection d'objets, la reconnaissance de formes et l'identification des personnes.
 - ✚ La planification : fournit des fonctionnalités pour la planification des tâches des robots, telles que la planification des mouvements, la planification des tâches et la planification de la production.
 - ✚ La commande : fournit des fonctionnalités pour la commande des robots, telles que la commande des mouvements, la commande des tâches et la commande de la production.

ROS2

Principaux packages

Principaux packages fournis par ROS2

- `rclcpp` et `rclpy` sont les packages qui implémentent les interfaces client en C++ et en Python pour ROS2
- `ros2cli` est le package qui fournit les outils en ligne de commande pour interagir avec ROS2. Il permet de lister, d'inspecter, de publier, de s'abonner, de faire des requêtes, de lancer, etc. les éléments de ROS2.
- `ros2launch` permet de lancer des applications ROS2 à partir de fichiers XML ou YAML. Il permet de définir les noeuds, les paramètres, les remappings, les variables d'environnement, etc. à lancer.
- `ros2control` fournit une interface unifiée pour contrôler différents types de robots. Il permet de définir des contrôleurs, des capteurs, des actionneurs, des transmissions, etc. et de les gérer à travers un gestionnaire de contrôle.
- `ros2navigation` fournit les algorithmes et les outils pour la navigation autonome des robots mobiles.
- `ros2vision` fournit les algorithmes et les outils pour la perception visuelle des robots.
- `ros2web` permet d'interfacer ROS2 avec le web. Il permet de créer des interfaces utilisateur à travers un navigateur web.

ROS2

Plan

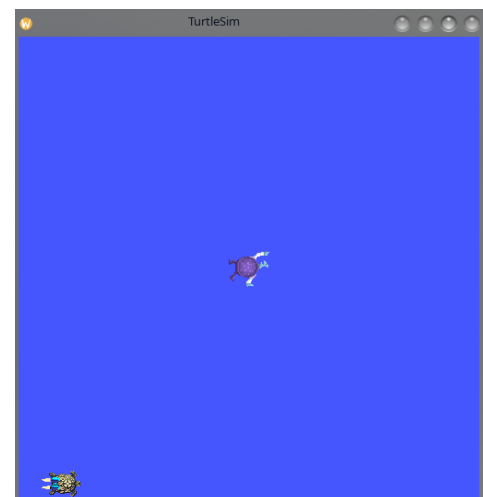
- 1 Introduction
- 2 Architectures
- 3 Outils de visualisation et de simulation
- 4 Interface CLI de ROS2

Turtlesim

L'outil de simulation : turtlesim

- ↳ Turtlesim est un petit simulateur pour apprendre ROS2
 - ▣ Contrôler une tortue virtuelle qui se déplace dans une fenêtre et dessine des formes avec son crayon
 - ▣ Créer plusieurs tortues, les faire interagir entre elles, et changer leur apparence et leur comportement
- ↳ Lancement de `turtlesim`

```
ros2 run turtlesim turtlesim_node
```



Turtlesim

L'outil de simulation turtlesim : les topics (1/2)

↳ Turtlesim fournit les trois topics suivants :

- 1 /turtle1/cmd_vel : ce topic publie les commandes de vitesse de la tortue. Les messages de ce topic sont de type geometry_msgs/Twist

```
1 # Vitesse linéaire
2 Vector3 linear
3 float64 x
4 float64 y
5 float64 z
6
7 # Vitesse angulaire
8 Vector3 angular
9 float64 x
10 float64 y
11 float64 z
```

- 2 /turtle1/color_sensor : publie les informations sur la couleur du trait laissé par la tortue

```
1 uint8 r
2 uint8 g
3 uint8 b
```

ROS2

Turtlesim

L'outil de simulation turtlesim : les topics (2/2)

- 3 /turtle1/pose : publie la position et la direction de la tortue. Les messages de ce topic sont de type geometry_msgs/Pose

```
1 # Position de la tortue
2 float32 x
3 float32 y
4 float32 theta
5
6 # Vitesse de la tortue
7 float32 linear_velocity
8 float32 angular_velocity
```

ROS2

Turtlesim

L'outil de simulation turtlesim : les services

↳ Turtlesim fournit plusieurs services, parmi lesquelles :

- ↳ `/clear` : efface l'écran de la simulation
- ↳ `/spawn` : crée une nouvelle tortue à une position donnée

```
1 float32 x
2 float32 y
3 float32 theta
4 string name #Optional. A unique name will be created and returned if this is empty
5 ---
6 string name
```

- ↳ `/reset` : réinitialise la position de la tortue à sa position initiale
- ↳ `/turtle1/set_pen` : modifie les propriétés du stylo utilisé pour dessiner

```
1 uint8 r
2 uint8 g
3 uint8 b
4 uint8 width
5 uint8 off
6 ---
```

ROS2

Turtlesim

L'outil de simulation turtlesim : les services

- ↳ `/turtle1/teleport_absolute` : téléporte la tortue à une position absolue donnée

```
1 float32 x
2 float32 y
3 float32 theta
4 ---
```

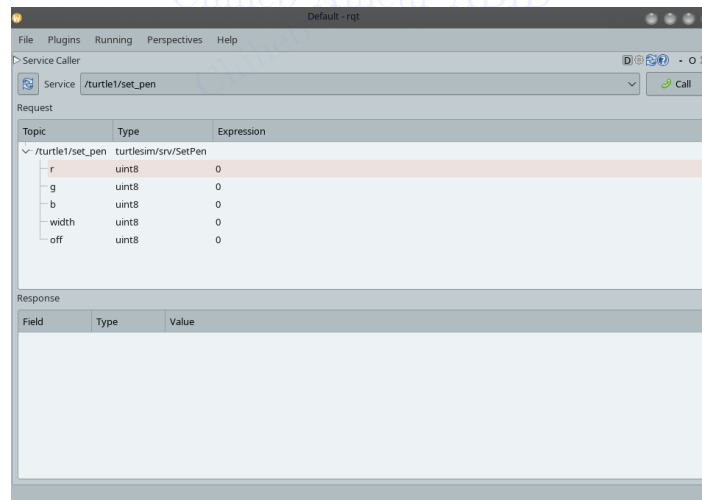
- ↳ `/kill` : tue une tortue donnée

```
1 string name
2 ---
```

ROS2

L'outil rqt

- ↳ rqt est une suite d'outils graphiques pour ROS2
- ↳ Elle fournit une interface utilisateur graphique (GUI) pour les tâches courantes de développement et de débogage d'applications ROS2
- ↳ Elle est basée sur un framework de plugins
 - ✎ De nouveaux plugins peuvent être ajoutés pour ajouter de nouvelles fonctionnalités
 - ✎ Il existe déjà un grand nombre de plugins disponibles, qui couvrent une large gamme de tâches

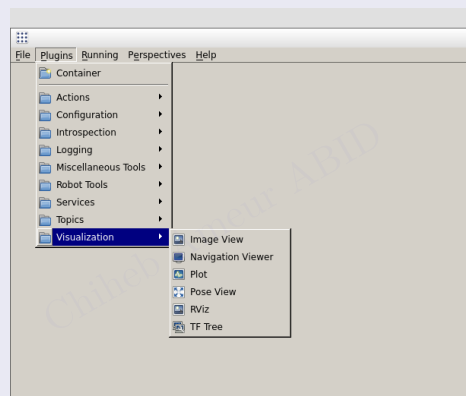


ROS2

L'outil rqt

L'outil rqt : plugins

- ↳ Les outils (plugins) fournis par rqt sont accessibles depuis le menu Plugins



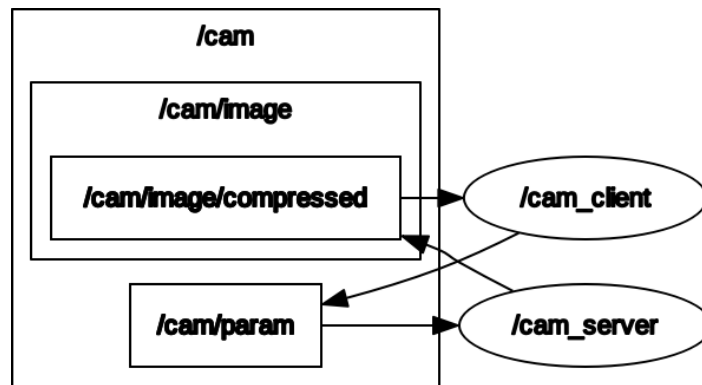
- ↳ Les principaux outils fournis par rqt :
 - ✎ rqt_graph permet de visualiser le graphe de calcul ROS d'une application
 - ✎ rqt_console permet d'afficher les messages ROS qui sont transmis entre les noeuds d'une application
 - ✎ rqt_param permet de configurer les paramètres ROS d'un système
 - ✎ rqt_node permet de gérer (démarrer, arrêter et redémarrer) les noeuds ROS2
 - ✎ rqt_debug permet de déboguer les applications ROS

ROS2

L'outil rqt

L'outil rqt : le plugin rqt_graph

- L'outil `rqt_graph` est un plugin de la suite d'outils `rqt` qui permet de visualiser le graphe de calcul ROS
- Le graphe de calcul ROS est un diagramme qui représente les relations entre les différents noeuds ROS qui composent une application ROS
- Principales fonctionnalités de `rqt_graph`
 - 👉 Visualiser le graphe de calcul ROS d'une application
 - 👉 Identifier les problèmes potentiels dans le graphe de calcul ROS
 - 👉 Modifier le graphe de calcul ROS



ROS2

L'outil Rviz2

Présentation

- Rviz2 est un outil de visualisation graphique 3D
 - 👉 Visualiser les données de leurs robots, capteurs et environnements
 - 👉 Il est basé sur Qt5, utilise OpenGL pour la visualisation et Ogre3D comme moteur de rendu 3D
- Rviz2 prend en charge un large éventail de données
 - 👉 La position et l'orientation des robots
 - 👉 Les données des capteurs, tels que les caméras, les lidars et les sonars
 - 👉 Les cartes et les modèles 3D
- Il offre divers types de visualisation
 - 👉 La superposition de données de différentes sources
 - 👉 L'animation de données
 - 👉 La personnalisation de l'apparence des données

ROS2

L'outil Rviz2

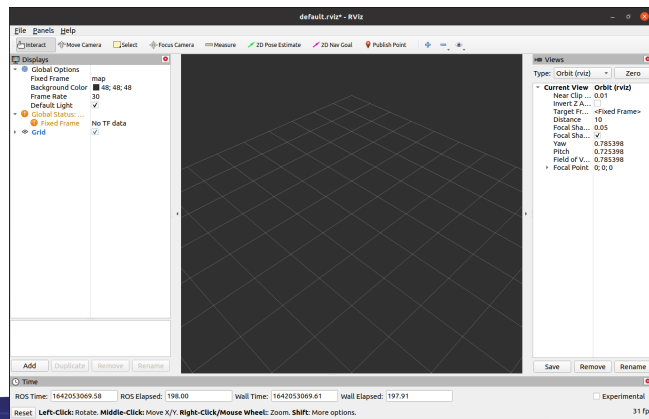
- ↳ Rviz2 reconnaît plusieurs modèles de description de robots, tels que :
 - ✎ URDF (Unified Robot Description Format) est un format open-source de fichiers XML pour décrire la géométrie et la cinématique des robots
 - ✎ SDF (Simulation Description Format) est un format similaire à URDF, mais plus récent. Il a été initialement développé comme partie du simulateur de robot Gazebo

↳ Lancer Rviz2

```

1 # Depuis le shell
2 rviz2
3 # Ou bien en utilisant la ligne de commande ROS2
4 ros2 run rviz2 rviz2

```



ROS2

Plan

- 1 Introduction
- 2 Architectures
- 3 Outils de visualisation et de simulation
- 4 Interface CLI de ROS2

ROS2

Interface CLI de ROS2

Présentation

- L'interface CLI de ROS2 est un ensemble de commandes qui permettent aux développeurs de contrôler et de gérer leurs applications ROS2 à partir de la ligne de commande.
- Les commandes de l'interface CLI peuvent être divisées en plusieurs catégories
 - ✚ Gestion des packages ; gérer les packages de ROS2
 - ✚ Démarrage et d'arrêt : démarrer et arrêter des applications ROS2.
 - ✚ Gestion des noeuds : gérer les noeuds ROS2, tels que la liste des noeuds en cours d'exécution, l'envoi de messages à des noeuds et la réception de messages de noeuds.
 - ✚ Gestion des topics : gérer les topics ROS2, tels que la liste des topics disponibles, la publication de messages sur des topics et la souscription à des topics.
 - ✚ Gestion des services : gérer les services ROS2, tels que la liste des services disponibles, l'appel de services et la publication de réponses à des services.
 - ✚ Gestion des paramètres : gérer les paramètres ROS2, tels que la liste des paramètres disponibles, la lecture de paramètres et l'écriture de paramètres.

ROS2

Interface CLI de ROS2

Présentation

- Syntaxe d'une commande ROS2

```
ros2 <command>
```
- Principaux commandes
 - ✚ `pkg` : gérer les packages
 - ✚ `topic` : gérer les topics
 - ✚ `run` : démarrer un noeud
 - ✚ `param` : gestion des paramètres

ROS2

Gestion des packages

Commandes de gestion des packages

↳ Gérer les packages

```
ros2 pkg <command>
```

✦ Créer un nouveau package

```
ros2 pkg create <package_name> [OPTIONS]
```

✦ Lister les exécutables

```
ros2 pkg executables [--full-path] [package_name]
```

✦ Lister les packages installés

```
ros2 pkg list
```

✦ Afficher l'emplacement d'un package

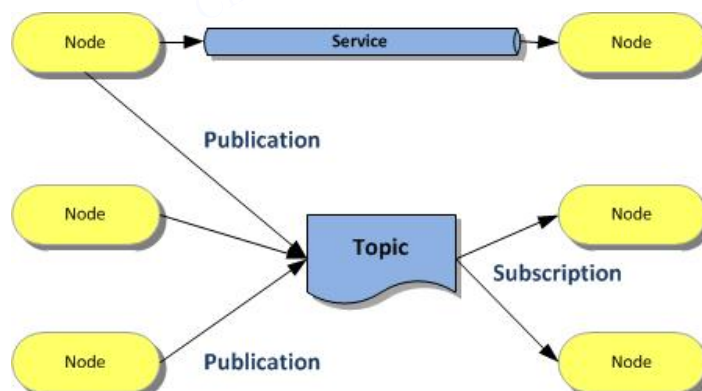
```
ros2 pkg prefix <package_name>
```

Interface CLI de ROS2

Les noeuds

↳ Définition

- ✦ Le processus associé à l'exécution d'un fichier exécutable dans un paquet ROS
- ✦ Les noeuds ROS utilisent une librairie client pour communiquer avec les autres noeuds. Les noeuds peuvent publier ou souscrire à des topics.
- ✦ Les noeuds peuvent fournir ou utiliser un service
- ✦ Les librairies client sont `rospy` pour python et `roscpp` pour C++.



Gestion des noeuds

Commandes de gestion des noeuds

➤ Un noeud est un processus associé à l'exécution d'un fichier exécutable dans un paquet

➤ Création d'un noeud

```
ros2 run <package_name> <executable_name>
```

➤ Montrer les noms de tous les nodes actifs

```
ros2 node list
```

➤ Obtenir des informations sur un noeud

```
ros2 node info <node_name>
```

Gestion des topics

Présentation

➤ Les topics sont des canaux de communication entre les noeuds. Ils permettent aux noeuds de publier et de s'abonner à des données.

➤ Lorsqu'un noeud publie des données sur un topic, il envoie les données à tous les noeuds qui sont abonnés à ce topic.

➤ Les topics ROS2 sont définis par les éléments suivants :

- ✎ Le nom du topic est une chaîne de caractères qui identifie le topic.
- ✎ Le type de données qui sont publiées sur le topic.
- ✎ L'espace de nommage est une chaîne de caractères qui permet de distinguer les topics de différents packages.

➤ ROS2 fournit les deux topics suivants à tous les noeuds :

- ① `/rosout` est la console de ROS2 contenant les messages de journalisation
- ② `/parameter_events` est utilisé pour publier des événements liés aux paramètres. Ces événements peuvent être utilisés par des noeuds pour être avertis des changements de paramètres

Gestion des topics

Présentation

- Les règles pour définir un topic ROS2 sont les suivantes :
 - ✎ Le nom du topic doit être une chaîne de caractères alphanumérique commençant par une lettre. Il ne doit pas contenir d'espaces ni de caractères spéciaux, à l'exception des underscores (_).
 - ✎ Le type de données doit être un message ROS2. Les messages ROS2 sont des structures de données qui définissent le format des données qui sont publiées sur le topic.
 - ✎ L'espace de nommage est facultatif. S'il est défini, il doit être une chaîne de caractères alphanumérique commençant par une lettre. Il ne doit pas contenir d'espaces ni de caractères spéciaux, à l'exception des underscores (_).

➤ Exemples de topics

```
1 /my_topic
2 /my_namespace/my_topic
3 /my_topic_with_underscores
4 /my_topic/my_message
```

Gestion des topics

Commandes de gestion des topics

- Les topics sont des données publiées par des noeuds et auxquelles les noeuds souscrivent
- Syntaxe

```
ros2 topic <cmd>
```
- Les commandes acceptées par `ros2 topic`
 - ✎ `bw <topic>` : Affiche la bande passante prise par le topic
 - ✎ `echo <topic>` : Affiche le contenu du topic en texte
 - ✎ `hz <topic>` : Affiche la fréquence de publication du topic
 - ✎ `info <topic>` : Fournit des informations sur un topic
 - ✎ `list` : Affiche la liste des topics actifs
 - ✎ `pub <topic> <msg_type> [values]` : Publie des données sur le topic
 - ✎ `type <topic>` : Affiche le type du topic

Gestion des topics

Publication des messages

↳ Syntaxe

```
ros2 topic pub <topic> <msg_type> [values] [OPTIONS]
```

- ✎ Les valeurs d'un message doivent être spécifiées au format YAML
- ✎ Si les valeurs ne sont pas spécifiées, l'envoi s'effectuera avec les valeurs par défaut

↳ Options

- ✎ `-r N` est le taux de publication en N Hz
- ✎ `-1, --once` publier un message, puis quitter

Gestion des topics

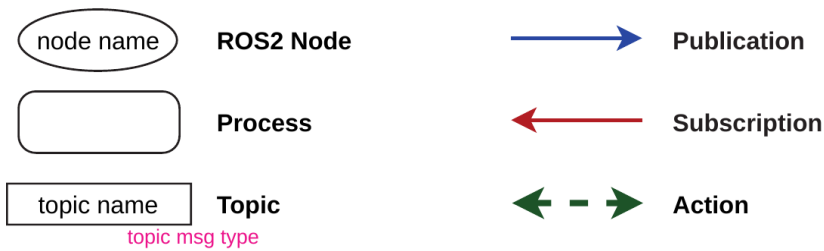
Exemple d'envoi d'un message

```
1 # Dans un premier terminal, on lance le noeud turtlesim
2 ros2 run turtlesim turtlesim_node
3
4 # Dans un deuxième terminal, on commence déterminer le type de messages du topic
5 ros2 topic type /turtle1/cmd_vel
6 # On affiche la structure d'un message
7 ros2 topic type /turtle1/cmd_vel | ros2 interface show -
8 # Publier un message
9 ros2 topic pub /turtle1/cmd_vel geometry_msgs/msg/Twist "linear: {x: 1}"
```

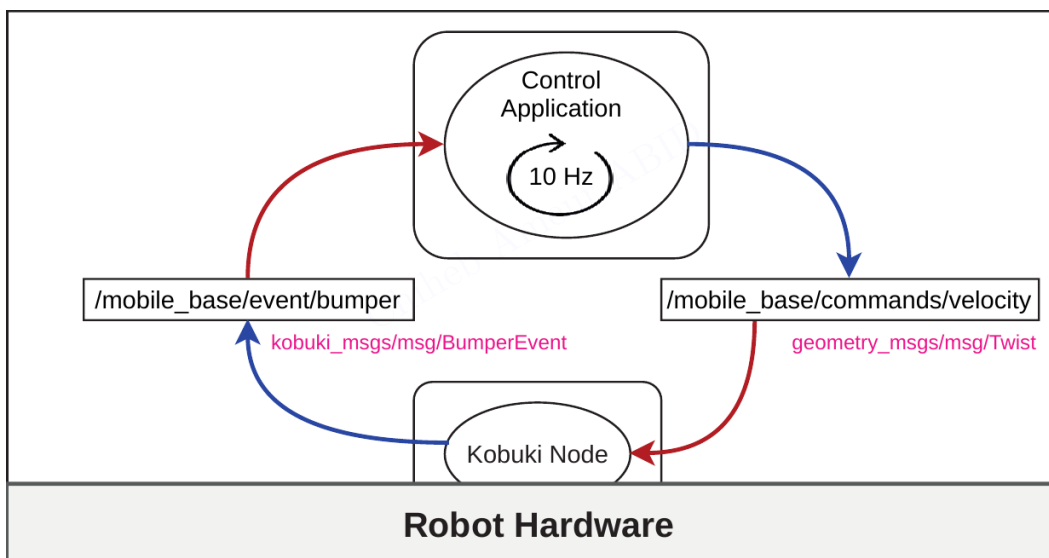
Graphe de calcul

Présentation

- Une abstraction puissante qui permet de modéliser de manière flexible les systèmes robotiques complexes
- Un graphe de calcul est un ensemble de noeuds et de liens qui représentent les dépendances entre ces noeuds
 - 🔗 Un noeud = un composant logiciel qui exécute des tâches
 - 🔗 Un lien = une liaison pour transmettre des données entre les noeuds



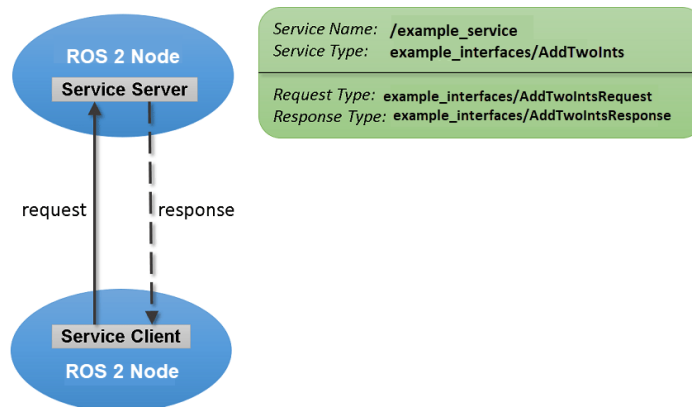
Exemple de graphe de calcul



Gestion des services

Présentation

- Les services sont des interfaces de programmation d'applications (API) qui permettent aux noeuds de ROS2 de communiquer entre eux de manière bidirectionnelle
- Un service ROS2 est composé de deux parties :
 - 👉 Le fournisseur de service est un noeud qui fournit un service
 - 👉 Le consommateur de service appelle le service en fournissant les paramètres d'entrée. Le fournisseur de service exécute le service et renvoie la réponse au consommateur de service



ROS2

Gestion des services

Commandes de gestion des services

- Les services sont une autre façon pour les noeuds de communiquer entre eux. Les services permettent aux noeuds d'envoyer des requêtes et de recevoir des réponses.
- Syntaxe

```
ros2 service <cmd>
```
- Les commandes acceptées par `ros2 service`
 - 👉 `call <service> [args]` : Appelle le service
 - 👉 `find <service_type>` : Trouve une liste de services disponibles pour un type spécifique
 - 👉 `list` : Liste des services actifs
 - 👉 `type` : Affiche le type d'un service

ROS2

Paramètres

Commandes de gestion des paramètres

- ROS-2 gère les paramètres qui permettent de stocker des données
 - ✚ Contrairement à ROS-1 les paramètres ne sont pas centralisés mais répartis sur chacun des nodes.
 - ✚ Ils servent à stocker : le modèle du robot (`robot_description`), des trajectoires de mouvements, des gains, etc.
 - ✚ Les paramètres peuvent se charger et se sauvegarder grâce au format YAML (Yet Another Language)
 - ✚ Ils peuvent également être définis en ligne de commande

Paramètres

Commandes de gestion des paramètres

- Commande de gestions des paramètres

```
ros2 param <cmd>
```

- Lister les paramètres disponibles

```
ros2 param list [OPTIONS] [node_name]
```

Options

```
--param-prefixes PARAM_PREFIXES [PARAM_PREFIXES ...] : montrer les paramètres avec les préfixes spécifiés
--param-type : Afficher les paramètres avec les types
```

- Charger les paramètres pour un noeud

```
ros2 param load node_name parameter_file [OPTIONS]
```

- Montrer les paramètres d'un noeud au format YAML

```
ros2 param dump node_name [OPTIONS]
```

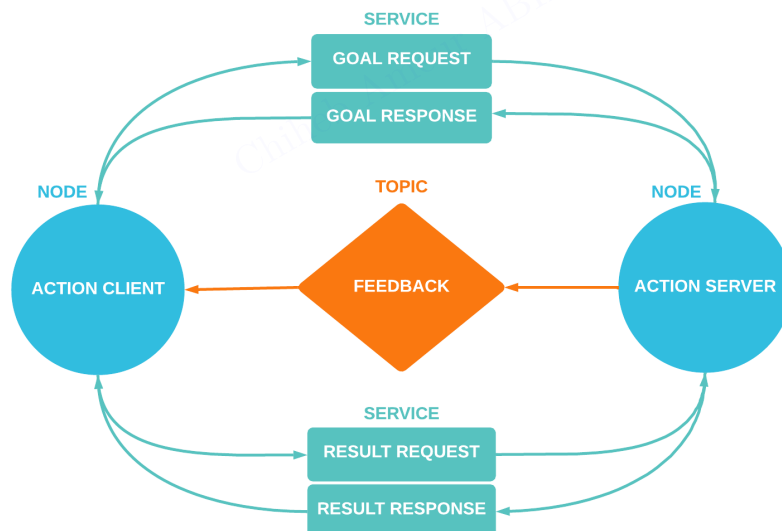
Actions

Principe de fonctionnement des actions (1/2)

- Les actions sont l'un des types de communications de ROS-2 et sont utilisées pour des tâches relativement longues
 - 👉 Elles sont constituées de trois parties : un but, un retour d'état, et un résultat.
 - 👉 Les actions sont construites sur les topics et les services. Leurs fonctionnalités sont similaires aux services, excepté que les actions peuvent être annulées.
 - 👉 Fournissent également un retour constant sur l'état à l'opposé des services qui ne retournent qu'une seule réponse.

Principe de fonctionnement des actions (2/2)

- Les actions utilisent un modèle client-serveur, similaire au modèle publisher-subscriber.
 - 👉 Un node "action client" envoie un but à un node serveur d'action "action server" qui accuse la réception du but et retourne un flux de retour d'état et un résultat



Actions

Commandes CLI de gestion des actions

- Afficher la liste des actions disponibles

```
ros2 action list [-t] [-c]
```

- 👉 -t : montrer le type de l'action
- 👉 -c : afficher uniquement le nombre d'actions

- Afficher des informations d'une action

```
ros2 action info <action_name> [-t] [-c]
```

- 👉 -t : montrer le type de l'action
- 👉 -c : afficher uniquement le nombre de serveurs et de clients de l'action

- Envoyer un objectif à une action

```
ros2 action send_goal <action_name> <action_type> <goal> [-f]
```

- 👉 -f : afficher le feedback pour l'objectif

Actions

Exemple d'utilisation des actions

- On utilise turtlesim

```
1 # Dans un premier terminal, on lance le noeud turtlesim
2 ros2 run turtlesim turtlesim_node
3
4 # Dans un deuxième terminal, on liste les actions disponibles avec leurs types d'
  interface
5 ros2 action list -t
6 # Afficher la structure de l'interface turtlesim/action/RotateAbsolute
7 ros2 interface show turtlesim/action/RotateAbsolute
8 # Envoyer l'objectif en activant le feedback avec l'option -f
9 ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "\xi
  theta: 3.14}" -f
```

Merci pour votre attention



Questions ?