UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Creación y uso de archivos launch en ROS2

Eugenio Ivorra

En esta práctica, se pretende explorar el uso y las posibilidades que ofrece los archivos launch para ejecutar sistemas complejos. Este conocimiento permitirá configurar y ejecutar múltiples nodos de la aplicación de forma simultánea sin necesidad de su ejecución en terminales distintas. Los objetivos específicos de este módulo son:

1. Familiarizarse con el proceso de creación y estructuración de un archivo launch en ROS 2.

2. Aprender a integrar múltiples nodos en un archivo launch, incluyendo el uso de parámetros, remapeos y configuraciones específicas.

3. Comprender y utilizar las funcionalidades de ROS 2 para cargar, configurar y modificar parámetros de nodos en tiempo de ejecución.

4. Aplicar los conceptos aprendidos para lanzar el nodo de turtlesim, el servidor de acciones y el cliente en un único archivo launch.

5. Evaluar y verificar la correcta comunicación entre los nodos lanzados utilizando herramientas como `rqt_graph`.

6. Analizar y reflexionar sobre las ventajas de utilizar archivos launch en sistemas ROS 2 complejos, identificando casos de uso y beneficios en la configuración, gestión y modificación de nodos y sistemas.

Índice general

1 Creación de archivos launch

1.1 Introducción

El sistema launch en ROS 2 es responsable de ayudar al usuario a describir la configuración de su sistema y luego ejecutarlo según lo descrito. La configuración del sistema incluye qué programas ejecutar, dónde ejecutarlos, qué argumentos pasarles, y convenciones específicas de ROS que facilitan la reutilización de componentes en todo el sistema, otorgándoles a cada uno una configuración diferente. También es responsable de monitorear el estado de los procesos lanzados, e informar y/o reaccionar ante cambios en el estado de esos procesos.

Los archivos launch escritos en Python, XML o YAML pueden iniciar y detener diferentes nodos, así como activar y actuar en diversos eventos. El paquete que proporciona este marco es `launch_ros`, que utiliza el marco `launch` no específico de ROS.

En la carpeta `/ROS2_examples/rclpy/launch/minimal_launch/launch/` se muestra el mismo archivo de lanzamiento implementado en Python, XML y YAML. Cada archivo de lanzamiento realiza las siguientes acciones:

- Configurar argumentos de línea de comando con valores predeterminados.
- Incluir otro archivo de lanzamiento.
- Incluir otro archivo de lanzamiento en otro espacio de nombres.
- Iniciar un nodo y configurar su espacio de nombres.
- Iniciar un nodo, configurar su espacio de nombres y configurar parámetros en ese nodo (utilizando los args).
- Crear un nodo para reasignar mensajes de un tópico a otro.

Estos archivo de lanzamiento están configurados para iniciar dos nodos turtlesim_node en espacios de nombres diferentes, con la opción de configurar el color de fondo del segundo nodo. También inicia dos instancias de un par de nodos talker y listener (de demo_nodes_cpp) y un nodo mimic para retransmitir mensajes entre los nodos turtlesim_node.

1.1.1 Propiedades Principales en un `Node` de ROS 2

Las siguientes propiedades representan las configuraciones más importantes que se pueden modificar en un 'Node' al utilizar un archivo 'launch.py' en ROS 2:

1. **package**: Nombre del paquete que contiene el ejecutable del nodo.
2. **executable**: Nombre del ejecutable del nodo.
3. **name**: Nombre con el que se ejecuta el nodo.
4. **namespace**: Espacio de nombres bajo el cual opera el nodo.
5. **parameters**: Lista o diccionario de parámetros que se pasarán al nodo.
6. **remappings**: Lista de remapeos de topics para el nodo.
7. **output**: Define a dónde se dirige la salida del nodo (por ejemplo, 'screen').
8. **log_level**: Nivel de registro del nodo ('DEBUG', 'INFO', 'WARN', etc.).
9. **env**: Variables de entorno específicas para el nodo.
10. **respawn**: Determina si el nodo debe reiniciarse si termina inesperadamente.

1.1.2 Python, XML o YAML: ¿Cuál debo usar?

Los archivos de lanzamiento en ROS 1 estaban escritos en XML, por lo que XML podría ser el formato más familiar para las personas que vienen de ROS 1. Para la mayoría de las aplicaciones, la elección del formato de lanzamiento en ROS 2 se reduce a la preferencia del desarrollador. Sin embargo, si tu archivo de lanzamiento requiere una flexibilidad que no puedes lograr con XML o YAML, puedes usar Python para escribir tu archivo de lanzamiento. Usar Python para el lanzamiento en ROS 2 es más flexible por las siguientes dos razones:

1. Python es un lenguaje de scripting, por lo que puedes aprovechar el lenguaje y sus bibliotecas en tus archivos de lanzamiento.
2. `ros2/launch` (características generales de lanzamiento) y `ros2/launch_ros` (características de lanzamiento específicas de ROS 2) están escritos en Python, por lo que tienes acceso de bajo nivel a las características de lanzamiento que pueden no estar expuestas por XML y YAML.

Dicho esto, un archivo de lanzamiento escrito en Python puede ser más complejo, dar más información y ser más legible que uno en XML o YAML.

1.2 Crear un archivo launch

Crea un nuevo directorio para guardar tus archivos launch en el paquete mi_paquete_python:

```
mkdir launch
```

1.2.1 Escribir el archivo launch

Vamos a crear un archivo launch de ROS 2 utilizando el paquete turtlesim y sus ejecutables. Como se mencionó anteriormente, esto puede ser en Python, XML o YAML pero en este caso concreto lo haremos en python.

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            executable='mimic',
            name='mimic',
            remappings=[
                ('/input/pose', '/turtlesim1/turtle1/pose'),
                ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel'),
            ]
        )
    ])
```

1.2.2 Examinar el archivo launch

Este archivo launch lanza un sistema de tres nodos, todos del paquete turtlesim. El objetivo del sistema es lanzar dos ventanas turtlesim, y hacer que una tortuga imite los movimientos de la otra. A continuación se detalla su funcionalidad:

1. **Importaciones**:
   - `LaunchDescription`: Esta es una clase que representa la descripción de un lanzamiento en ROS 2. Define qué nodos y acciones se iniciarán.
   - `Node`: Esta es una acción que permite lanzar un nodo ROS 2.
2. **Función** `generate_launch_description()`:
   - Esta función es necesaria en todos los archivos de lanzamiento de ROS 2 escritos en Python. Su propósito es generar y devolver la descripción de lo que se debe lanzar.
3. **Nodos** `turtlesim_node`:
   - Se lanza un nodo llamado `sim` del paquete `turtlesim` con el ejecutable `turtlesim_node` en el espacio de nombres `turtlesim1`.
   - Se lanza otro nodo con las mismas características pero en el espacio de nombres `turtlesim2`. Por ende, habrá dos simulaciones de tortugas corriendo en espacios de nombres diferentes.
4. **Nodo** `mimic`:
   - Se lanza un nodo llamado `mimic` del paquete `turtlesim` con el ejecutable `mimic`.
   - Este nodo tiene remapeos (`remappings`) configurados. El propósito de estos remapeos es redirigir ciertos tópicos. En este caso:
     - Los mensajes de `/input/pose` serán redirigidos a `/turtlesim1/turtle1/pose`.
     - Los mensajes de `/output/cmd_vel` irán a `/turtlesim2/turtle1/cmd_vel`.
5. **Retorno**:
   - La función devuelve una `LaunchDescription` que agrupa los tres nodos para ser lanzados cuando se ejecuta este archivo de lanzamiento.

En resumen, este archivo de lanzamiento inicia dos simulaciones de tortugas en espacios de nombres diferentes (`turtlesim1` y `turtlesim2`) y un nodo `mimic` que redirige ciertos mensajes entre las dos simulaciones.

1.2.3 Configuración del paquete

Abre `package.xml` y añade lo siguiente:

```
  <depend>launch</depend>
  <depend>launch_ros</depend>
```

La primera línea indica que el paquete tiene una dependencia en el paquete launch. El paquete launch es el paquete principal para el sistema de lanzamiento en ROS2 y proporciona funcionalidades para ejecutar múltiples nodos y otras acciones durante el inicio de un sistema de robots. La segunda línea es similar al anterior, pero es específico para ROS. Añade funcionalidades para lanzar nodos ROS, cargar parámetros, establecer remapeos, etc. Ambas dependencias son esenciales si quieres utilizar ficheros .launch.py para lanzar nodos ROS desde tu paquete.

Abre `setup.py` y añade lo siguiente:

```
from setuptools import setup
import os
from glob import glob
...
    data_files=[
        ...
        (os.path.join('share', package_name), glob('launch/*.launch.py'))
    ],
...
```

Estas modificaciones permiten a un paquete de ROS 2 utilizar el sistema de lanzamiento y aseguran que los archivos de lanzamiento se instalen en la ubicación correcta cuando el paquete se construye e instala. Ten en cuenta que el fichero launch tiene que tener esa terminación en concreto.

1.2.4 Ejecución del launch

Para ejecutar el archivo launch creado anteriormente, entra en el directorio donde lo creaste y ejecuta el siguiente comando:

```
ros2 launch turtlesim_mimic_launch.py
```

Es posible ejecutar un archivo launch directamente (como hacemos arriba) o proporcionado por un paquete cuando se compila. Compila tu paquete desde la raiz de tu workspace y luego llama al launch utilizando esta sintaxis:

```
ros2 launch <nombre_del_paquete> <nombre_del_archivo_launch>
```

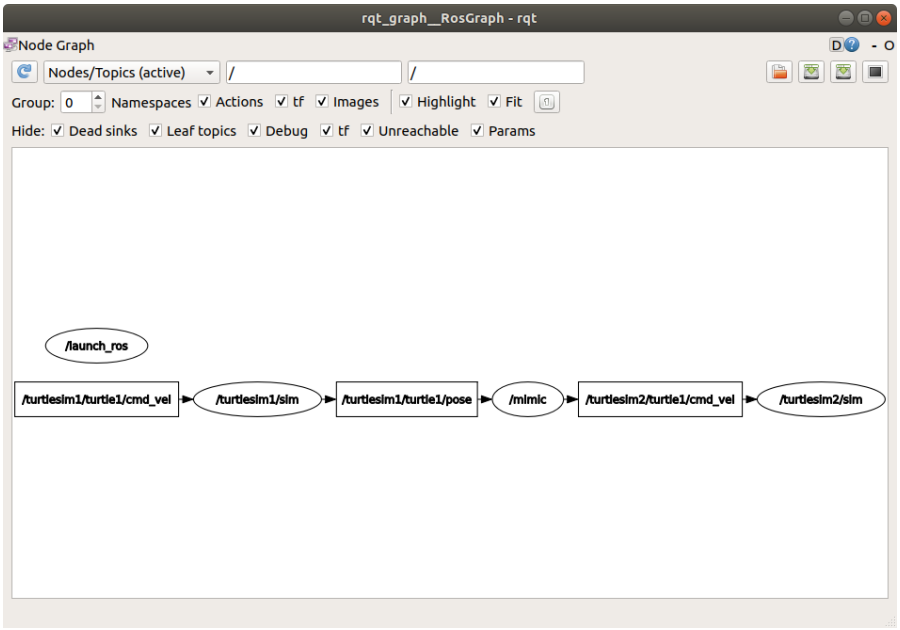

Figura 1.1: RQT_graph tras ejecutar el launch

1.2.5 Inspección con el sistema con `rqt_graph`

Cuando se tiene sistemas de ROS2 complejos o se lanzan muchos nodos es conveniente inspeccionar como se comunican entre sí a traves de la herramienta `rqt_graph`. Mientras el sistema aún está en ejecución, abre una nueva terminal y ejecuta `rqt_graph` para tener una mejor idea de la relación entre los nodos en tu archivo launch (figura 1.1).

```
rqt_graph
```

Los archivos launch simplifican la ejecución de sistemas complejos con muchos nodos y detalles de configuración específicos. Puedes crear archivos launch usando Python, XML o YAML y ejecutarlos usando el comando `ros2 launch`.

1.3 Actividad

Crea un launch que lance tus nodos de la práctica 6. Es decir, que ejecute el nodo de turtlesim, tu servidor de acciones y tu cliente para conseguir que la tortuga haga un círculo.

Bibliografía

Web de los tutoriales de ROS (2023). https://docs.ros.org/en/humble/Tutorials.html.

Web de ROS2 Humble (2023). https://docs.ros.org/en/humble/index.html.