18.ROS2 time related API

1. Introduction to time-related API

The time-related APIs of ros2 include Rate, Time, Duration, Time and Duration operations, etc., which are explained below.

2. Rate

In addition to timers, ROS2 also provides the Rate class, through which the running frequency of the program can also be controlled;

The Rate object in rclpy can be created through nodes. The sleep() function of the Rate object needs to be executed in a child thread, otherwise it will block the program.

Example: Periodically output a piece of text.

```
import rclpy
import threading
from rclpy.timer import Rate

rate = None
node = None

def do_some():
    global rate
    global node
    while rclpy.ok():
        node.get_logger().info("hello ---------")
        # hibernation
        rate.sleep()

def main():
    global rate
    global node
    rclpy.init()
    node = rclpy.create_node("rate_demo")
    # Create a Rate object
    rate = node.create_rate(1.0)

    # Create subthread
    thread = threading.Thread(target=do_some)
    thread.start()

    rclpy.shutdown()

if __name__ == "__main__":
    main()
```

3.Time

Example: Create a Time object and call its functions.

```
import rclpy
from rclpy.time import Time
def main():
    rclpy.init()
    node = rclpy.create_node("time_demo")
    # Create Time object
    right_now = node.get_clock().now()
    t1 = Time(seconds=10,nanoseconds=500000000)

    node.get_logger().info("s = %.2f, ns = %d" % (right_now.seconds_nanoseconds()
[0], right_now.seconds_nanoseconds()[1]))
    node.get_logger().info("s = %.2f, ns = %d" % (t1.seconds_nanoseconds()[0],
t1.seconds_nanoseconds()[1]))
    node.get_logger().info("ns = %d" % right_now.nanoseconds)
    node.get_logger().info("ns = %d" % t1.nanoseconds)
    rclpy.shutdown()

if __name__ == "__main__":
    main()
```

4.Duration

Example: Create a Duration object and call its function.

```
import rclpy
from rclpy.duration import Duration

def main():
    rclpy.init()

    node = rclpy.create_node("duration_demo")
    du1 = Duration(seconds = 2,nanoseconds = 500000000)
    node.get_logger().info("ns = %d" % du1.nanoseconds)

    rclpy.shutdown()

if __name__ == "__main__":

    main()
```

5. Time and Duration operations

Example: Time and Duration related operations.

```
import rclpy
from rclpy.time import Time
from rclpy.duration import Duration
```

```
def main():
    rclpy.init()
    node = rclpy.create_node("time_opt_node")
    t1 = Time(seconds=10)
    t2 = Time(seconds=4)

    du1 = Duration(seconds=3)
    du2 = Duration(seconds=5)

    # Compare
    node.get_logger().info("t1 >= t2 ? %d" % (t1 >= t2))
    node.get_logger().info("t1 < t2 ? %d" % (t1 < t2))
    # Math operations
    t3 = t1 + du1
    t4 = t1 - t2
    t5 = t1 - du1

    node.get_logger().info("t3 = %d" % t3.nanoseconds)
    node.get_logger().info("t4 = %d" % t4.nanoseconds)
    node.get_logger().info("t5 = %d" % t5.nanoseconds)

    # Compare
    node.get_logger().info("-" * 80)
    node.get_logger().info("du1 >= du2 ? %d" % (du1 >= du2))
    node.get_logger().info("du1 < du2 ? %d" % (du1 < du2))

    rclpy.shutdown()

if __name__ == "__main__":
    main()
```