Introduction to ROS2: Basics, Motion, and Vision

INNOPOLIS UNIVERSITY

ROS Debugging

1. In general, we can use **gdb** for debugging. Also, memory leaks can be checked with **valgrind**. To see the graphical graph representation and performance analysis, **callgrind** and **KCachegrind**.
2. There are two ways you can debug your code

to debug with gdb

```
cd <path to workspace>/install/ros2_visualize/lib/ros2_visualize
gdb joy_hagen_sub
> (gdb) r
```

However, this way **we can not load parameters** and other config files.

ROS Debugging

1. Second option is to define the gdb options under launch-prefix the launch file

to debug with gdb

```
apt-get install xterm
prefix=['xterm -e gdb -ex run –args']
```

Depending on the situation, you can use different keys to control your program execution flow. r for running the program, then bt for backtracing if you got into some problems.

ROS Debugging: Memory Handling

1. If there is problems with memory leaks or with performance, valgraid can be used

to debug with valgrind

```
sudo apt-get install valgrind
sudo apt install kcachegrind
prefix=['valgrind –tool=callgrind –dump-instr=yes -v –instr-atstart=no'],
```

2. The above prefix does not start callgrind right after it stars, i.e., instr-atstart=no

to start debugging with valgrind

```
callgrind_control -i on
```

ROS Debugging: Callgrind + Kcachegrind

1. Callgrind is a profiling tool that records the call history among functions in UNIX process

to locate dump file

```
locate callgrind.out
```

for visualizing the profile

```
kcachegrind <path/some>/callgrind.out.xxxx
```

ROS Logging

1. Logging is usually performance-wise expensive,
2. However, log4cxx, which is a port of log4j logger library, has a null footprint on performance
3. Logging has several LEVELs: DEBUG, INFO, WARN, ERROR, and FATAL
4. Different between ERROR and FATAL is that if there is an error, but program can still run, logging should be defined as ERROR otherwise FATAL

to import logging functionality

```
#include <rclcpp/rclcpp.hpp>
#include <rclcpp/logger.hpp>
#include <rcutils/error_handling.h>
```

ROS Logging

1. How can we do logging?

to define logging

RCLCPP_<LEVEL>[_<OTHER>], e.g., RCLCPP_INFO("INFO message");
RCLCPP_<LEVEL>[_STREAM]_<OTHER>, e.g.,
RCLCPP_<LEVEL>[_STREAM]_COND[_NAMED]
RCLCPP_INFO_STREAM_COND(val < 0., "INFO stream message; val (" « val « ") < 0");
RCLCPP_INFO_STREAM_ONCE("INFO stream message");