How to setup Ubuntu user to be able to run a real-time node:

1. Editing the configuration file

If your system has no directory called /etc/security/limits.d then you will need to edit /etc/security/limits.conf. If /etc/security/limits.d does exist on your machine, then you will need to create and edit a file called /etc/security/limits.d/rtgroup.conf. The file must contain (at least) the following two lines:

```
@rtgroup   -  rtprio     95

@rtgroup   -  memlock    unlimited
```

Contrary to a lot of misinformation on the web, there is no reason to include a line here that provides enhanced “niceness” control, which is completely irrelevant for realtime scheduling and low latency applications.

2. Creating an “rtgroup” group

As the super-user (“root”) run the following commands from a terminal window:

```
groupadd rtgroup

usermod -a -G rtgroup yourUserID
```

You should substitute your actual user id or “login” for “yourUserID”.

For these changes to take effect, “yourUserID” must log out and log in again.

Where to Find an Example Real-Time Code for ROS2

In the same folder, where this file is, you can find the following files:

publisher_member_function.cpp

subscriber_member_function.cpp

publisher_member_function_RT.cpp

subscriber_member_function_RT.cpp

The first two are standard ROS2 nodes in C++. The second two show how to convert the standard ROS2 node in C++ to a real-time node by adding extra functions.

The extra functions were found in a 4-part series of posts, starting at the web address below:

https://shuhaowu.com/blog/2022/01-linux-rt-appdev-part1.html

Making Sure Hyper-Threading is off!

2.1. How to Know if We Have Hyper-Threading Enabled?

There are two techniques we can use to check if our system's using HTT. We can call lscpu and just search (with grep) on its output the number of threads that each core has:

```
$ lscpu | grep "Thread(s) per core"

Thread(s) per core: 2
```

From this output, we know that HTT is enabled because there's more than one thread per core (two in this case). In case we had only one thread per core, we would know that HTT is disabled.

We can also inspect the file that contains the flag with the activation mode of SMT:

```
$ cat /sys/devices/system/cpu/smt/active

1
```

A value of 1 indicates that we have HTT enabled, while a 0 indicates that we have it disabled.

How to Disable RT Throttling

Before the widespread availability of multicore systems, if an RT process uses up all of the available CPU time, it can cause the entire system to hang. This is because the Linux scheduler will not run a non-RT process if the RT process continuously hogs the CPU. To avoid this kind of system lockup, especially on desktop-oriented systems where any process can request to be RT, the Linux kernel has a feature to throttle RT processes if it uses 0.95 s out of every 1 s of CPU time. This is done by pausing the process for the last 0.05 s and thus may result in deadline misses during the moments when the process is paused.

This can be turned off by writing the value -1 to the file /proc/sys/kernel/sched_rt_runtime_us on every system boot.

How to compile a ROS2 package for the fastest run

Use:

```
colcon build --cmake-arg -DCMAKE_BUILD_TYPE=Release –packages-select <package-name>
```