IN325: 2 – RT threads

Author : Christophe Garion <garion@isae.fr>
Audience : 3A IN
Date :

Abstract

The objectives of this lab session is to manipulate real-time threads and use the JamaicaVM RTSJ compliant JVM.

In all the following exercises, when executing Java application either through the standard JVM or the Jamaica VM, use the `taskset` utility to use only one processing unit. For instance:

```
taskset -c 0 java MyExample
```

1 Priority inversion example

Implement:

- a `EatCPUTime` class with a static method `eat(long millisecond)` that does silly computations for millisecond ms (use the static method `currentTimeMillis` of the `System` class to compute elapsed time).
- a `Logger` class to store strings inside a `ArrayList`. Use a static attribute and static methods.
- a `Lock` class with a synchronized method `acquireLock` that calls the `eat` method for 5s. When a thread enters and exits this method, some messages about the thread will be added to the log.
- a `AcquireLockRunnable` class implementing `Runnable` whose `run` method use the `acquireLock` method on a `Lock` object. Messages when the `run` has started and is about to finish will be added to the log.
- a `DummyRunnable` class implementing `Runnable` whose `run` method calls the `eat` method for 2s. Messages when the `run` has started and is about to finish will be added to the log.

The messages will use the thread name with `getName`.
Create a program that:

- create a `Lock` object `l`
- create a thread `t1` using `AcquireLockRunnable` with priority `MIN_PRIORITY`
- create a thread `t2` using `AcquireLockRunnable` with priority `MAX_PRIORITY`
- create a thread `t3` using `DummyRunnable` with priority `NORM_PRIORITY`
- start `t1`, then `t2`, then `t3`, wait for the 3 threads to finish and print the logs.

What happens? Is it normal if you think about POSIX SCHED_FIFO policy?

2 Verifying JamaicaVM installation

Verify that JamaicaVM is correctly installed:

1. print the value of the `JAMAICA_HOME` environment variable:

   ```
   echo $JAMAICA_HOME
   ```

   If the variable is not set, find the installation location of Jamaica (e.g. with `ls -ld /usr/local/jamaica*` and add the following lines to your `.bashrc` file:

   ```
   export JAMAICA_HOME=/usr/local/jamaica-where-you-found-it
   export PATH=$PATH:$JAMAICA_HOME/bin
   ```

Start a new terminal to take your modifications into account or source .bashrc with the following command:

```
source ~/.bashrc
```

2. start the aicas license provider with the -ping to verify that aicas servers are online:

```
aicasLicenseProvider -ping
```

   In order to be able to use the tools of the JamaicaVM Personal Edition, you have to start the license provider in a separate shell or in background. JamaicaVM tools will contact the license provider in order to work. You will have to provide your own license key to enable them.

3. copy the directory $JAMAICA_HOME/target/linux-x86/examples/HelloWorld and launch the Ant tool in this directory:

```
cd HelloWorld
ant run
```

   Normally, execution of the Ant target should produce no errors.

3 Priority inversion: is it now correct?

Use real-time thread and FIFO scheduling policy for the JamaicaVM (by setting the environment variable JAMAICA_SCHEDULING to FIFO, see [1]), verify that the priority inversion problem is solved.
Use the Thread Control tool to see the threads activity. In order to do that, start the JamaicaVM with the following option:

```
jamaicavm -Djamaica.scheduler_events_port=2712
```

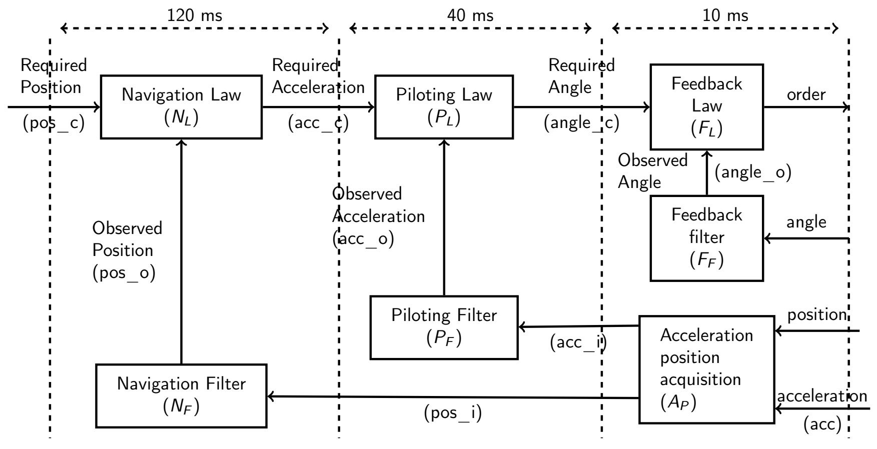
and start the Thread Monitor with the following command:

```
jamaica_threadmonitor
```

Use [1] as a reference for the tool.

4 A Flight Control System

This exercise is taken from an exercise developed by Éric Noulard. Many thanks to him.
We consider a simplified Flight Control System represented on figure 1.



Task	T	C	r	D
NL	120	20	0	120
NF	120	10	0	120
PL	40	5	0	40
PF	40	5	0	40
FL	10	2	0	10
FF	10	1	0	10
AP	10	1	0	10

Figure 1: A simplified Flight Control System

A rate monotonic analysis of this set of tasks will show that the task set is feasible (on a single processor host) and would give the following priorities (the higher the better):

task name	priority
AP	7
FF	6
FL	5
PF	4
PL	3
NF	2
NL	1

In your repository, you will find under the `lab2` directory a classical Java application called `RMScheduling` which implements the previous simplified control system. Each periodic task is represented by a `Periodic` object which is a `java.lang.Runnable`.

1. read and understand the `Periodic` and the `RMScheduling` classes.
2. compile and execute the `RMScheduling` application several times using a number of cycles of 100. What do you observe? Do not forget to use `taskset` to use only one processor.
3. transform this application into a `RTRMScheduling` application in order to solve the previous problems.
4. run the `RTRMScheduling` application. What do you observe?
5. use the `jamaicabuilder` application to create an executable (do not forget to use the `-schedulingPolicy` option to set the scheduling policy, cf. [1]). Execute the application. What do you observe?

References

[1] aicas GmbH. *JamaicaVM 6.2 - User Manual*. Available in `$JAMAICA_HOME/doc`, 2013.