Yarp Threads Exercise

Carlos Beltran-Gonzalez

October 22, 2007

1 Introduction

In this exercise the student should be able to understand completely what a YARP thread is. The student should use the YARP documentation to study the classes and the library structure. The examples section, in particular, an example with YARP thread should be used by the student to develop the work.

Exercise goal

Create a program that starts a number of threads that write messages in the standard output (terminal).

2 Download the documentation

Download the yarp documentation using the command:

```
wget http://eris.liralab.it/yarp/specs/dox/yarpdoc.zip
```

In the case the Internet connections is not available the instructor will distribute the zip file using a pen drive.

2.1 Decompress

Decompress the documentation in your preferred directory:

```
unzip yarpdoc.zip
```

Access the documentation by using the command:

```
firefox index.html
```

The instructor will explain the structure of the YARP documentation and how to navigate through the documentation structure

3 Exercise 1

Now, the student should create the threads program. It is requested to create program that creates three threads, starts them, sleeps for a second, and then terminates the threads. Each thread must write in the standard output an initialization message with its number. The threads must be created with an initialization number. Next you can find a pseudocode of the exercise:

```
#include<stdio.h>
#include<yarp/os/Thread.h>
#include<yarp/os/Time.h>

class MyThread: public Thread
{
    ...
    public:
        virtual bool threadInit()
        {
            ...
        }
        virtual void run()
        {
            while(!isStopping())
            {
               ...
            }
        }
        ......
};

int main (....)
{
    // Here declare your threads
    ....
    // Here start your thread
    ...
    // Here add a delay for the main loop
    ...
```

```
        // Here stop your threads
        ...
        return 0;
    }
```

TIP: Use previous code

It is recommended to reuse the code development in the previous exercise (hello world example) and rename the necessary parts.

Check the output

When the program has been compiled and running, check the output and ask the instructor to see your code.

4 Exercise 2

Make the same thing that before, but this time the class MyThread must reside in another file. Write also the corresponding headers file (MyThread.h). Modify the CMakeLists.txt to include the new file and modify the code thus everything compiles and runs nicely as before.

5 Exercise 3 (optional)

Make as before, but this time create and destroy the Threads dynamically. Use some sort of YARP data structure to store the references(pointer) to the dynamically created thread's.

6 Exercise 4 (optional)

Concurrent Programming: Make as before but use a YARP classes to protect the common access of threads to the standard output. Eventually, create a common data structure and protect it with the YARP classes. *TIP!!* A YARP class used to manage concurrent access to process resources is: yarp::os::Semaphore

References

[1] Yarp Open source project, *Yet Another Robotic Platform*, 2004

[2] Bruce Eckel, *Thinking in C++*, Prentice Hall.