CST8244 – Real-Time Programming

Lab 6 – Namespace

Introduction

To successfully create a communications channel on QNX's Neutrino RTOS, you require the following pieces of information:

- the **N**ode **D**escriptor
- the server's **P**rocess **ID**
- the server's **CH**annel **ID**

These three pieces of information are known as the: ND/PID/CHID

Unfortunately, two pieces of information ---- the PID and CHID --- are not known until run-time, when the server application has been launched. This is not scalable, and makes it difficult to create a resource manager, which is QNX's version of a device driver on other operating systems, such as Linux.

A better solution, and one that is more scalable, is to make use of QNX's namespace. To implement this strategy, make the following changes to Phase I of message passing: server calls the name_attach() function to register the name in the namepace and to create a channel. Next, the client calls the name_open() function to open the named server connection.

Phase II requires the fewest changes. The same three functions are called:

- client sends message to client: MsgSend()
- server received message from client: MsgReceive()
- server replies to client: MsgReply()

The only change is to MsgReceive(): you'll need to deference the variable of type *name_attach_t to get the channel ID, which is the function's first parameter.

In Phase III, the server calls the name_detach() function to remove the name from the namespace and destroy the channel, and the client calls the name_close() function to close a server connection that was opened by name_open().

Refactor the Door-Entry-System (DES) to use QNX's Namespace

Every computer scientist needs to know how to refactor code. In the context of software development, refactoring code means to improve the code without introducing new functional requirements.

In this lab, you're to refactor your solution to Assignment #1 – Door Entry System (DES) such that the inputs, controller and display programs no longer rely on the ND/PID/CHID, and instead make use of QNX's namespace**.**

Running the Programs

In PuTTY session #1, startup the display to see the human friendly output: **# des_display**

In PuTTY session #2, startup the controller: **# des_controller**

In VMware's console window, run the inputs program using file re-direction: **# des_inputs < exitLefttDoor.txt**

where *exitLeftDoor.txt* is a text file that contains:

```
rs
54321
gru
ro
ws
100
rc
grl
glu
lo
lc
gll
exit
```

The run-time behaviour of DES should be **exactly** the same when compared to the run-time behaviour of the non-namespace version. As well, all three programs should gracefully terminate on the *exit* input event.

Marking Scheme

The lab is marked out of 10 points.

- 2 marks for **des_display**
  - calls name_attach() to register a name in the namespace and to create a channel to receive messages from the controller
  - calls name_detach()
- 3 marks for **des_controller**
  - Calls name_open() to open a name for the server connection managed by display
  - calls name_attach() to register a name in the namespace and to create a channel to receive messages from inputs
  - calls name_close()
  - calls name_detach()
- 2 marks for **des_inputs**
  - calls name_open() to open a name for the server connection managed by the controller
  - calls name_close()
- 3 marks for demonstration by screen-shot

Special note if you did not complete Assignment #1: for a B grade (i.e. 7/10), refactor your Lab #5 – Message Passing to use QNX's namespace.

Special note if you did not complete Assignment #1, nor did you complete Lab #5: for a C grade (i.e. 6/10), refactor the reference example demonstrating message passing. The reference example can be found on Brightspace.

Deliverables

Before making the zip-file deliverable (next item), please action:

- Format your source code to make it easier for me to read. Momentics IDE will do this for you: Source -> Format
- Verify your projects build cleanly. Please action: a) Project -> Clean… and b) Project -> Build All

Prepare a zip-file that contains the following items:

1. Take a single, composite screen-shot of your Desktop showing the following scenario:
   PuTTY #1: display
   PuTTY #2: controller
   VMware Console window: inputs < exitLeftDoor.txt
2. Source code for the display program.
3. Source code for the controller program.
4. Source code for the inputs program.
5. Name your zip-file according to your Algonquin College username:
   cst8244_lab6_yourUsername.zip

   For Example, cst8244_lab6_bond0007.zip
6. Upload and submit your zip-file to Brightspace before the due date.