**QNX Introductory Project**

1. **Task Objectives**

Develop 1 server application and 2 client applications for QNX that communicate via IPC and perform basic operations:

* + math operations with 32-bit signed integers
    1. add 2 numbers
    2. subtract 2 numbers
    3. multiply 2 numbers
    4. divide 2 numbers
  + string operations (strings with length between 1 and 16 characters)
    1. concatenate 2 strings
    2. search for substring within a string (return the position of the substring)

You need to select a design that will allow easy change of communication media. Two sets of media need to be use in separate applications:

* + shared memory
  + mq

1. **Implementation Guidelines**
   1. **Communication Library**

This includes:

* + defining a format for data transfers over the shared memory communication channel;

Implementing synchronous (add, subtract) and asynchronous (multiply, divide, concatenate, search) function calls over the communication channel to the clients;

* + synchronizing the access to the common memory between the processes (use mutexes, semaphores, named semaphores, conditional variables);
  + using separate threads for each message processing (one thread for each command).
  1. **Server Application**

The server must use the communication library to exchange requests and responses with his clients. The server implementation must include separate thread for different command types (1 thread for math & 1 thread for string operations). It is not forbidden the server to run more threads than these two but it must implement at least them.

* 1. **Client Applications**

The 2 client applications must provide the following functionality:

* + the 1st client application must implement the operations “add 2 numbers”, “multiply 2 numbers” and “concatenate 2 strings”;
  + the 1st client application must use the shared library directly (rely on the process loader for loading & linking);
  + the 2nd client application must implement the operations “subtract 2 numbers”, “divide 2 numbers” and “search for substring within a string”;

The 2 applications must provide a simple command line user interface for demonstrating the functionality.

These applications can be demonstrated in a text console with a simple command-line user interface:

For client 1

|  |
| --- |
| 1. **Add 2 numbers** 2. **Multiply 2 numbers** 3. **Concatenate 2 strings** 4. **Exit**   **Enter command: 1**  **Enter operand 1: 42**  **Enter operand 2: 37**  **Sending request…**  **Receiving request…**  **Result is 79!** |

Another use case for Client 1 could be (can be simulated with a big delay in the server while processing two numbers multiplication and a small or no delay while processing concatenation):

|  |
| --- |
| 1. **Add 2 numbers** 2. **Multiply 2 numbers** 3. **Concatenate 2 strings** 4. **Exit**   **Enter command: 2**  **Enter operand 1: 7**  **Enter operand 2: 8**  **Sending request (7\*8) …**   1. **Add 2 numbers** 2. **Multiply 2 numbers** 3. **Concatenate 2 strings** 4. **Exit**   **Enter command: 3**  **Enter operand 1: a**  **Enter operand 2: b**  **Sending request concat(a,b) …**   1. **Add 2 numbers** 2. **Multiply 2 numbers** 3. **Concatenate 2 strings** 4. **Exit**   **Enter command:**  **Receiving request from concat(a,b) command**  **Result from concat(a,b) is ab!**  **Receiving request from (7\*8) command**  **Result from (7\*8) is 56!** |

For client 2 following use case can be referred:

|  |
| --- |
| * + 1. **Subtract 2 numbers**     2. **Divide 2 numbers**     3. **Find substring in a string**     4. **Exit**   **Enter command: 3**  **Enter substring: lo**  **Enter string: Hello!**  **Sending request…**  **Receiving request…**  **Result is : 3 (if substring was not found print NULL)** |