Com S 454/554 Fall 2021 Assignment 2 (Due: Friday October 1)

Review the concepts and programming practice related to RPC (Remote Procedure Call), including the example/tutorial and the supplementary guidelines on parameter/argument types at the end of this description. Then, develop a Client/Server-based integer storage system as specified below.

Requirement: Development of a Simple Client/Server System

In this assignment, you are required to develop a distributed client/server system by utilizing RPC as communication middleware.

The server provides the following procedures (functions) to the clients:

- SUM(argument: a list of double-type numbers) – return the sum of the numbers;
- MAX(argument: a list of double-type numbers) – return the maximum of the numbers;
- MIN(argument: a list of double-type numbers) – return the minimum of the numbers.

The length of number list is not fixed.

With accesses to the above functions, each client should provide the following user interface:

- when a client is launched, it accepts the following inputs as command-line arguments:
    - hostname or IP-address of the server;
    - type of request, which is a upper-case string of value “SUM”, “MAX” or “MIN”;
    - the parameters
        - a list of double-type numbers with various length;
    - multiple parameters are separated by a blank;
    - when a function can’t be performed, print out appropriate error message (for example, the request name is invalid, the number list is empty, etc.).

This assignment should be written in C/C++. Each submission should include a **zipped package that contains all the source files that can be compiled with “make” to produce a server and a client program**.

Your programs should compile and runnable **on the same node** of cluster pyrite.cs.iastate.edu.

Score allocation:

1. Your programs can be compiled on pyrite.cs.iastate.edu – 20%; if cannot be compiled, receive 0.
2. User interface meets the requirement – 10%
3. The length of number list can be arbitrary – 10%
4. The 3 functions (SUM, MAX, MIN) are implemented correctly - 20%

5. IDL file – 10%
6. Server skeleton -10%
7. Client code – 10%
8. You have reasonably comment your code – 10%

Appendix: Supplementary information on parameter types in RPC

Declarations of Arrays

The rpcgen compiler lets you declare arrays within structures, unions, and typedefs. Arrays can be of fixed-length or of variable-length. Fixed length arrays are declared the same as their C language counterparts. Consider the following:

```
int proc_hits[100];
```

This declares proc_hits to be an array of 100 integers. The rpcgen compiler transfers this declaration into the header file without modification.

You can also specify variable length arrays. The maximum size of the array is enclosed in angle braces. You can omit the size to indicate that there is no maximum value. For example, consider the following declaration:

```
long x_coords<50>;
```

This declares x_coords to be an array of 0 to 50 long integers. Similarly, the following declaration:

```
long z_coords<>;
```

declares z_coords to be an array of long integers. In this case, the array size can be any value that can fit in an unsigned integer. Since the C language does not support variable-length arrays, the rpcgen compiler translates these declarations into a C data structure. The structure has a length indicator and a pointer to the array. For example, the declaration:

```
typedef long x_coords<50>;
```

gets translated into:

```
typedef struct {

    u_int x_coords_len;

    long *x_coords_val;

} x_coords;
```

The length indicator is an unsigned integer, and is formed by appending _len to the declaration name. The pointer to the array is formed by appending _val to the declaration name. When you implement the remote procedures, you must use the x_coords_len and the x_coords_val elements of the x_coords structure. When populating the structure, you must set the length of the array in the x_coords_len element and allocate space to store the array in the x_coords_val element.