МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)
Институт интеллектуальных кибернетических систем
Кафедра Кибернетики

Лабораторная работа №4

Выполнил студент группы Б17-501: Борзенков А.В.

Проверил: Ктитров С.В.

5

Москва, 2020

Задача:

Вариант D–3

Тема «Сериализация и десериализация»

Разработать программы, синхронизирующие содержимое структур данных в оперативной памяти, состоящих из полей целых (8, 16, 32, 64-разрядных) и вещественных (float, double) чисел. Взаимодействующие (по сети) программы могут быть запущены на компьютерах разных архитектур.

Код программы:

data.x:

```
1. struct data {
2.     char char_arg;
3.     short short_arg;
4.     int int_arg;
5.     long long_arg;
6.     float float_arg;
7.     double double_arg;
8. };
```

data.h:

```
1. /*
2.  * Please do not edit this file.
3.  * It was generated using rpcgen.
4.  */
5.
6. #ifndef _DATA_H_RPCGEN
7. #define _DATA_H_RPCGEN
8.
9. #include <rpc/rpc.h>
10.
11.
12.   #ifdef __cplusplus
13.   extern "C" {
14.   #endif
15.
16.
17.   struct data {
18.           char char_arg;
19.           short short_arg;
20.           int int_arg;
21.           long long_arg;
22.           float float_arg;
23.           double double_arg;
24.   };
25.   typedef struct data data;
26.
```

```
27.   /* the xdr functions */
28.
29.   #if defined(__STDC__) || defined(__cplusplus)
30.   extern  bool_t xdr_data (XDR *, data*);
31.
32.   #else /* K&R C */
33.   extern bool_t xdr_data ();
34.
35.   #endif /* K&R C */
36.
37.   #ifdef __cplusplus
38.   }
39.   #endif
40.
41.   #endif /* !_DATA_H_RPCGEN */
```

data_xdr.c:

```
1. /*
2.  * Please do not edit this file.
3.  * It was generated using rpcgen.
4.  */
5.
6. #include "data.h"
7.
8. bool_t
9. xdr_data (XDR *xdrs, data *objp)
10.   {
11.           register int32_t *buf;
12.
13.
14.           if (xdrs->x_op == XDR_ENCODE) {
15.                   if (!xdr_char (xdrs, &objp->char_arg))
16.                           return FALSE;
17.                   buf = XDR_INLINE (xdrs, 3 * BYTES_PER_XDR_UNIT);
18.                   if (buf == NULL) {
19.                           if (!xdr_short (xdrs, &objp->short_arg))
20.                                   return FALSE;
21.                           if (!xdr_int (xdrs, &objp->int_arg))
22.                                   return FALSE;
23.                           if (!xdr_long (xdrs, &objp->long_arg))
24.                                   return FALSE;
25.
26.                   } else {
27.                   IXDR_PUT_SHORT(buf, objp->short_arg);
28.                   IXDR_PUT_LONG(buf, objp->int_arg);
29.                   IXDR_PUT_LONG(buf, objp->long_arg);
30.                   }
31.                    if (!xdr_float (xdrs, &objp->float_arg))
32.                           return FALSE;
```

```
33.                     if (!xdr_double (xdrs, &objp->double_arg))
34.                             return FALSE;
35.                     return TRUE;
36.             } else if (xdrs->x_op == XDR_DECODE) {
37.                     if (!xdr_char (xdrs, &objp->char_arg))
38.                             return FALSE;
39.                     buf = XDR_INLINE (xdrs, 3 * BYTES_PER_XDR_UNIT);
40.                     if (buf == NULL) {
41.                             if (!xdr_short (xdrs, &objp->short_arg))
42.                                     return FALSE;
43.                             if (!xdr_int (xdrs, &objp->int_arg))
44.                                     return FALSE;
45.                             if (!xdr_long (xdrs, &objp->long_arg))
46.                                     return FALSE;
47.
48.                     } else {
49.                     objp->short_arg = IXDR_GET_SHORT(buf);
50.                     objp->int_arg = IXDR_GET_LONG(buf);
51.                     objp->long_arg = IXDR_GET_LONG(buf);
52.                     }
53.                     if (!xdr_float (xdrs, &objp->float_arg))
54.                             return FALSE;
55.                     if (!xdr_double (xdrs, &objp->double_arg))
56.                             return FALSE;
57.             return TRUE;
58.             }
59.
60.             if (!xdr_char (xdrs, &objp->char_arg))
61.                     return FALSE;
62.             if (!xdr_short (xdrs, &objp->short_arg))
63.                     return FALSE;
64.             if (!xdr_int (xdrs, &objp->int_arg))
65.                     return FALSE;
66.             if (!xdr_long (xdrs, &objp->long_arg))
67.                     return FALSE;
68.             if (!xdr_float (xdrs, &objp->float_arg))
69.                     return FALSE;
70.             if (!xdr_double (xdrs, &objp->double_arg))
71.                     return FALSE;
72.             return TRUE;
73.     }
```

client/client.c

```
1. #include <arpa/inet.h>
2. #include <unistd.h>
3. #include <stdio.h>
4. #include <stdlib.h>
5. #include <stdint.h>
6.
```

```
7. #include "data.h"
8. #include "write.h"
9.
10.   #define ADDRESS "127.0.0.1"
11.   #define PORT 10001
12.   #define BUFFSIZE 8192
13.
14.
15.   extern String GetStateInXDR(data*);
16.
17.   int socket_des;
18.   struct sockaddr_in server_addr;
19.
20.
21.   int main() {
22.       /* init */
23.       {
24.           if((socket_des = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
25.               perror("Socket error: ");
26.               exit(EXIT_FAILURE);
27.           }
28.
29.           server_addr.sin_family = AF_INET;
30.           server_addr.sin_port = htons(PORT);
31.           inet_aton(ADDRESS, &server_addr.sin_addr);
32.       }
33.
34.       data my_data;
35.       /* data init */{
36.           my_data.char_arg = 1;
37.           my_data.short_arg = 2;
38.           my_data.int_arg = 3;
39.           my_data.long_arg = 4;
40.           my_data.float_arg = 5.;
41.           my_data.double_arg = 6.;
42.       }
43.
44.
45.       for (int itr = 0; itr < 5; itr++) {
46.           String currentState = GetStateInXDR(&my_data);
47.           int msg_size = currentState.len;
48.           uint8_t test_msg[msg_size];
49.           for (int i = 0; i < msg_size; i++) {
50.               test_msg[i] = currentState.str[i];
51.           }
52.
53.           if (sendto(socket_des, test_msg, msg_size, 0, (struct
    sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
54.               perror("Send error:");
55.               exit(EXIT_FAILURE);
```

```
            }
            // change data
            my_data.char_arg++;
            my_data.int_arg++;
            my_data.long_arg *= 2;
        }

        close(socket_des);
        exit(EXIT_SUCCESS);
    }
```

string.h

```
#ifndef MY_STRING
#define MY_STRING

#include <stdlib.h>

struct String {
     char* str;
     int len;
};

   typedef struct String String;

   struct String CreateString(const char* s, int len) {
        String result;
        result.str = malloc(len);
        for (int i = 0; i < len; i++) {
            result.str[i] = s[i];
        }
        result.len = len;
        return result;
   }

   #endif

```

write.h

```
#include <stdio.h>
#include <unistd.h>

#include "string.h"
#include "data.h"

#define BUFFSIZE 8192

String GetStateInXDR(data* out) {
            XDR xhandle;
```

```
11.       char* buff = malloc(BUFFSIZE);
12.           u_int size;
13.       xdrmem_create( &xhandle, buff, BUFFSIZE, XDR_ENCODE );
14.       if ( xdr_data( &xhandle, out ) != TRUE ) {
15.           printf("Error\n");
16.       }
17.       size = xdr_getpos( &xhandle );
18.       return CreateString(buff, size);
19.   }
```

server/server.h

```
1. #include <sys/socket.h>
2. #include <resolv.h>
3. #include <unistd.h>
4. #include <stdint.h>
5. #include <stdlib.h>
6. #include <stdio.h>
7.
8. #include "read.h"
9.
10.   #define PORT 10001
11.   #define BUFFSIZE 8192
12.
13.   uint8_t msg[BUFFSIZE];
14.
15.
16.   int main() {
17.       /* init */
18.       int msg_size, socket_des;
19.       struct sockaddr_in server_addr, client_addr;
20.       int addr_size = sizeof(client_addr);
21.       {
22.           if((socket_des = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
23.               perror("socket error: ");
24.               exit(EXIT_FAILURE);
25.           }
26.
27.           server_addr.sin_family = AF_INET;
28.           server_addr.sin_port = htons(PORT);
29.           server_addr.sin_addr.s_addr = INADDR_ANY;
30.
31.           if(bind(socket_des, (struct sockaddr*)&server_addr,
   sizeof(server_addr)) != 0) {
32.               perror("bind error: ");
33.               exit(EXIT_FAILURE);
34.           }
35.       }
36.
37.
```

```
38.       data my_data;
39.       /* data init */
40.       {
41.           my_data.char_arg = 0;
42.           my_data.short_arg = 0;
43.           my_data.int_arg = 0;
44.           my_data.long_arg = 0;
45.           my_data.float_arg = 0;
46.           my_data.double_arg = 0;
47.       }
48.
49.       while (1) {
50.           msg_size = recvfrom(socket_des, msg, sizeof(msg), 0, (struct
   sockaddr*)&client_addr, &addr_size);
51.           char* buffer = malloc(msg_size);
52.           for (int i = 0; i < msg_size; i++) {
53.               buffer[i] = msg[i];
54.           }
55.           String cur = CreateString(buffer, msg_size);
56.
57.           my_data = GetDataFromXDR(cur);
58.
59.           printf("My data is: %d %d %d %lld %f %f\n"
60.                   , my_data.char_arg
61.                   , my_data.short_arg
62.                   , my_data.int_arg
63.                   , my_data.long_arg
64.                   , my_data.float_arg
65.                   , my_data.double_arg);
66.           if (my_data.char_arg == 5) {
67.               break;
68.           }
69.       }
70.
71.       close(socket_des);
72.       exit(EXIT_SUCCESS);
73.   }
```

server/read.h

```
1. #include <stdio.h>
2. #include <unistd.h>
3.
4. #include "data.h"
5. #include "string.h"
6.
7. #define BUFFSIZE 8192
8.
9. data GetDataFromXDR(String input) {
10.       XDR xhandle;
```

```
        data in;
        char* buff = malloc(BUFFSIZE);
        int n = input.len;
        for (int i = 0; i < n; i++) {
            buff[i] = input.str[i];
        }
        xdrmem_create( &xhandle, buff, n, XDR_DECODE );

        memset( &in, 0, sizeof(in) );
        if (xdr_data( &xhandle, &in ) != TRUE ) {
            printf("Error\n");
        }
        xdr_free(xdr_data, (char*)&in);

        return in;
    }
```

Makefile

```
SHELL = /bin/bash
run:
	gcc -c client/client.c -o client/client.o
	gcc -c client/data_xdr.c -o client/data_xdr.o
	gcc -c server/server.c -o server/server.o
	gcc -c server/data_xdr.c -o server/data_xdr.o
	gcc -o client/client client/client.o client/data_xdr.o
	gcc -o server/server server/server.o server/data_xdr.o
	./server/server
```

Результат выполнения: