# BITS Pilani - Hyderabad Campus
# Advanced Operating Systems

## Lab Assignment-3

## 1st Semester 2025-26

### Creating a Sun RPC Application.

**Step 1:** Install **rpcbind** and **libtirpc-dev** packages.

write the following command in the terminal:
***sudo apt-get install rpcbind libtirpc-dev***

```
$ sudo apt-get install rpcbind libtirpc-dev
```

**Step 2:** Creating the IDL file **(square.x)**

An IDL is a file (suffixed with .x) which optionally begins with a bunch of type definitions and then defines the remote procedures. In this program we have two type definitions to define a structure that holds one long int, this will be our input parameter for the square function. Our interface will also have one version and one program. We have to assign a number to each function, version, and program. The function will be given an ID of 1. So will be the version. The program number is a 32-bit number. Sun reserved the range from 0 to 0x1fffffff. We will number this program 0x13451111.

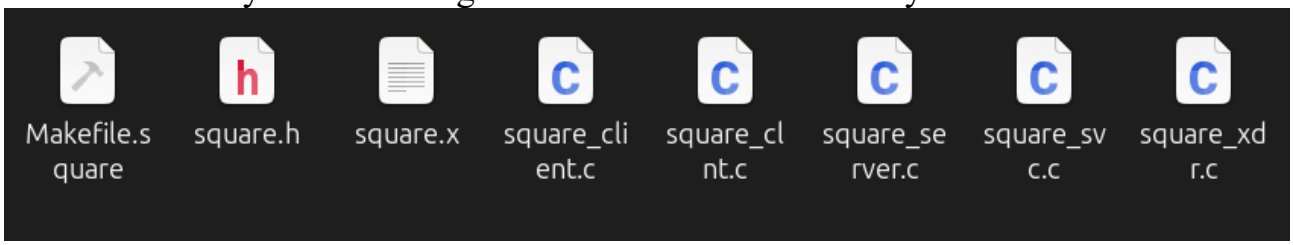
```
struct square_in {
        long arg1;
};
struct square_out {
        long res1;
};
program SQUARE_PROG {
        version SQUARE_VERS {
                square_out SQUAREPROC(square_in) = 1;
        } = 1;
} = 0x13451111;
```

Run the RPC generator “rpcgen” to generate client stub (square_clnt.c), server stub (square_svc.c), header file (square.h), and data coversion file (square_xdr.c)

command: ***rpcgen -a -C square.x***

```
)$ rpcgen -a -C square.x
```

All the necessary files will be generated in the same directory.




**Step 3:** Configuring flags in **Makefile.sqaure**
Open the Makefile.square file and edit the following flags.

```
CFLAGS += -g -I/usr/include/tirpc
LDLIBS += -ltirpc
```

**Step 4:** Creating server and client code.

**// server**
Open square_server.c and edit the code as required.

```
#include "square.h"

square_out *
squareproc_1_svc(square_in *argp, struct svc_req *rqstp)
{
	static square_out  result;

	result.res1 = argp -> arg1 * argp -> arg1;

	return &result;
}
```

**// client**
Open square_client.c and edit the code as required.

```
#include "square.h"


void
square_prog_1(char *host, long arg1)
{
	CLIENT *clnt;
```

```
        square_out  *result_1;
        square_in  squareproc_1_arg;

#ifndef        DEBUG
        clnt = clnt_create (host, SQUARE_PROG, SQUARE_VERS, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
#endif         /* DEBUG */
      //WRITE YOUR CODE HERE

      squareproc_1_arg.arg1 = arg1;
        result_1 = squareproc_1(&squareproc_1_arg, clnt);
        if (result_1 == (square_out *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        else {
          printf("Result: %ld\n", result_1 -> res1);
        }
#ifndef        DEBUG
        clnt_destroy (clnt);
#endif          /* DEBUG */
}


int
main (int argc, char *argv[])
{
        char *host;

        if (argc != 3) {
                printf ("usage: %s server_host <integer>\n", argv[0]);
                exit (1);
        }
        host = argv[1];
        square_prog_1 (host, atoi(argv[2]));
exit (0);
}
```

**Step 5:** Creating output files
command: ***make -f Makefile.square***

```
shashank@HP17K:~/Desktop/BPHC/AOS_LABWORK/SQ$ make -f Makefile.square
cc -g -I/usr/include/tirpc   -c -o square_clnt.o square_clnt.c
cc -g -I/usr/include/tirpc   -c -o square_client.o square_client.c
cc -g -I/usr/include/tirpc   -c -o square_xdr.o square_xdr.c
cc -g -I/usr/include/tirpc    -o square_client  square_clnt.o square_client.o square_xdr.o -ltirpc
cc -g -I/usr/include/tirpc   -c -o square_svc.o square_svc.c
cc -g -I/usr/include/tirpc   -c -o square_server.o square_server.c
cc -g -I/usr/include/tirpc    -o square_server  square_svc.o square_server.o square_xdr.o -ltirpc
```

It will generate all the ouput files including **square_server** and **square_client.**

**Step 6:** Run Server and Client on same machine, pass the value from client to the server and the server would return the square of the value as result.

```
shashank@HP17K:~/Desktop/BPHC/AOS_LABWORK/SQ$ ./square_server
```

```
shashank@HP17K:~/Desktop/BPHC/AOS_LABWORK/SQ$ ./square_client localhost 3
Result: 9
shashank@HP17K:~/Desktop/BPHC/AOS_LABWORK/SQ$
```

**Step 7:** The **rpcinfo -p** command shows each RPC-based service with port numbers, an RPC program number, a version number, and an IP protocol type (TCP or UDP). It can be used to get the port number through which the client is connected on the server side.

```
shashank@HP17K:~/Desktop/BPHC/AOS_LABWORK/SQ$ rpcinfo -p
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
 323293457    1   udp  53791
 323293457    1   tcp  65168
```

**Practice Questions:**

You have to implement a calculator service using remote procedure call (SUN RPC Package) as discussed, which takes 2 integer arguments as inputs and performs the following operations:

1. Addition of two numbers.

2. Multiplication of two numbers.
3. Subtraction of two numbers.
4. Division of two numbers.
5. Remainder of two numbers (First Number % Second Number).
6. Check if the first number (argument) is a prime number (For example, if the input is 23 and 10, then output should be 1 (or yes)).