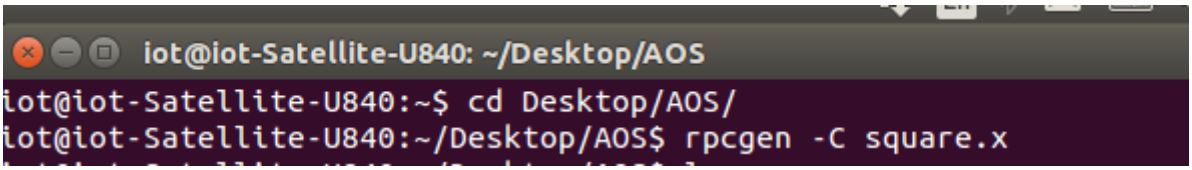
SUN RPC Program assignment

Creating a Sun RPC Application

Step 1. 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 definition 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 progam. We have to assign a number to each function, version, and program. The function will be given an ID of 1. So will the version. The program number is a 32-bit number. Sun reserved the range from 0 to 0x1ffffff. We'll 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)



Step 2. Creating client and server code

```
/* client.c */
#include<stdlib.h>
#include<stdio.h>
#include"square.h"
int main (int argc, char **argv)
{
CLIENT *cl;
square_in in;
square_out *out;
if (argc != 3) {
printf("client <localhost> <integer>");
exit (1);
}cl = clnt_create (argv[1], SQUARE_PROG, SQUARE_VERS,"tcp");
in.arg1 = atol(argv [2]);
if ((out = squareproc_1(&in, cl)) == NULL)
{
printf("Error\n");
exit(1);
}
```

```
printf("Result %ld\n",out->res1);
exit(0);
}
```

```
/* server.c */
#include "square.h"
#include <stdio.h>
square_out *squareproc_1_svc (square_in *inp, struct svc_req *rqstp)
{
static square_out outp;
outp.res1 = inp->arg1 * inp->arg1;
return (&outp);
}
```

```
iot@iot-Satellite-U840:~/Desktop/AOS$ gcc -lnsl -o client client.c square_clnt.c square_xdr.c
iot@iot-Satellite-U840:~/Desktop/AOS$ gcc -lrpcsvc -lnsl -o server server.c square_svc.c square_xdr.c
iot@iot-Satellite-U840:~/Desktop/AOS$
```

Step 3. 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.

```
iot@iot-Satellite-U840: ~/Desktop/AOS
iot@iot-Satellite-U840:~/Desktop/AOS$ ./server
```

```
iot@iot-Satellite-U840: ~/Desktop/AOS
iot@iot-Satellite-U840:~/Desktop/AOS$ ./client localhost 4
Result 16
```

Now you can try to run Server and Client on different machines.
Step 4.
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.

ASSIGNMENT (Total 10 marks):

Create a SUN RPC program in ubuntu system for finding factorial of a number.What are the purpose of the extra files which gets generated after running the “rpcgen” program? [2 mark]

Fill in the blanks after finishing and running your code in the system,each fill in blanks contains marks given in the bracket.Please *do not fill in the blanks without first doing in your system as programs output might be checked in random basis.*

filename: factorial.h

```
struct factorial_in
{
long int arg1;
};

struct factorial_out
{
long int res1;
};

program FACT_PROG{
        version FACT_VERS{
            __________ FACTORIALPROC(__________) = 1;   [0.5+0.5 = 1 mark]
        }=1;
}=0x13451111;
```

filename: client.c (i.e the client program)

```
#include<stdlib.h>
#include<stdio.h>
#include"_________"    [0.5 mark]


int main (int argc, char **argv)
{
CLIENT *cl;
factorial_in in;
factorial_out *out;
if (argc != 3) {
printf("client <localhost> <integer>");
exit (1);
}

cl = clnt_create (argv[1], FACT_PROG, _________,"tcp");  [0.5 mark]
in.arg1 = atol(argv [2]);
```

```
if ((_____)     [ 1 mark]
{
printf("Error\n");
exit(1);
}
printf("Result %ld\n",_________);   [1  mark]
exit(0);
}
```

filename: server.c (server file)

```
#include "_________" [0.5 mark]
#include <stdio.h>

factorial_out *factorialproc_1_svc (___________ *inp, struct svc_req *rqstp) [1 mark]
{
static __________ outp;  [0.5 mark]
int i;
i = inp->__________; [0.5 mark]
outp.res1 = 1;
while(i !=0)
{
outp.res1 = ____________;  [1 mark]
i--;
}
return (__________);  [0.5 mark]
}
```