Prerequisite for RPC

1. sudo apt install libntirpc-dev

2. dpkg -L libntirpc-dev

3. Need to symbolic link files so C preprocessor can find them, here is whole list,

sudo ln -s /usr/include/ntirpc/rpc/rpc.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/misc/abstract_atomic.h /usr/include/misc/

sudo ln -s /usr/include/ntirpc/netconfig.h /usr/include/misc/

sudo ln -s /usr/include/ntirpc/misc/stdio.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/intrinsic.h /usr/include

sudo ln -s /usr/include/ntirpc/rpc/tirpc_compat.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/auth.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/rpc_err.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/clnt_stat.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/auth_stat.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/clnt.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/misc/rbtree.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/misc/opr.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/misc/wait_queue.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/misc/queue.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/reentrant.h /usr/include

sudo ln -s /usr/include/ntirpc/rpc/svc.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/rpc_msg.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/work_pool.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/pool_queue.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/misc/portable.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/misc/timespec.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/misc/os_epoll.h /usr/include/misc

sudo ln -s /usr/include/ntirpc/rpc/auth_unix.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/rpcb_clnt.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/rpcb_prot.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/rpcent.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/types.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/netconfig.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/netconfig.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/netconfig.h /usr/include

sudo ln -s /usr/include/ntirpc/rpc/xdr.h /usr/include/rpc

sudo ln -s /usr/include/ntirpc/rpc/netconfig.h /usr/include

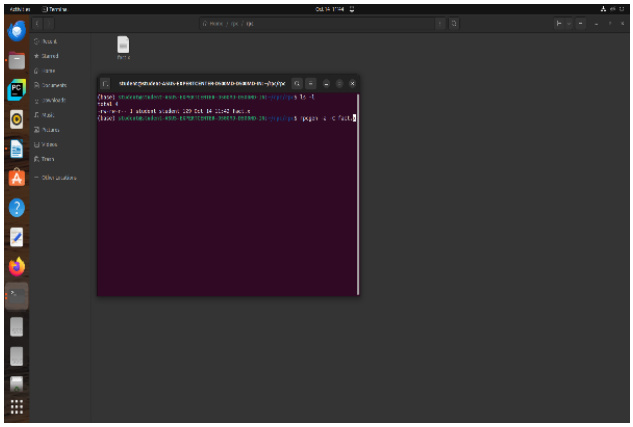
sudo ln -s /usr/include/ntirpc/rpc/xdr.h /usr/include

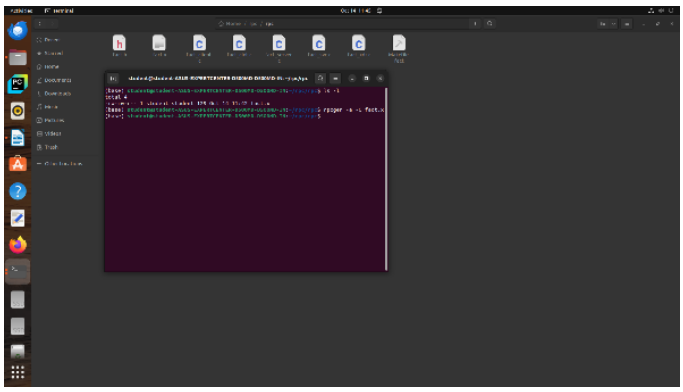
Steps to run RPC program:

1. Make a directory using following command in terminal: mkdir rpc
2. Open rpc directory: cd rpc
3. Create a fact.x file (Interface Definition language file): touch fact.x
4. Write the following code in fact.x file: gedit fact.x

```
struct intpair
{
        int a;
};

program FACT_PROG
{
version FACT_VERS
{
        int FACT(intpair) = 1;

} = 1;

} = 0x22222222;
```

5. To generate all the necessary template files to run RPC program: rpcgen -a -C fact.x





6. **We need to edit Makefile.fact template file for compiling the program:** gedit Makefile.fact
   and then to compile the program following command is used: make -f Makefile.fact
7. **Edit the Sever template file:** gedit fact_server .c
8. **Edit the Client template file:** gedit fact_client .c
9. **Again compile the program:** make -f Makefile.fact

```
/* Output of the Complinling the makefile.fact
cc -g -I/usr/include/tirpc   -c -o fact_clnt.o fact_clnt.c
cc -g -I/usr/include/tirpc   -c -o fact_client.o fact_client.c
cc -g -I/usr/include/tirpc   -c -o fact_xdr.o fact_xdr.c
cc -g -I/usr/include/tirpc    -o fact_client  fact_clnt.o fact_client.o fact_xdr.o -lnsl -ltirpc -ltirpc
cc -g -I/usr/include/tirpc   -c -o fact_svc.o fact_svc.c
cc -g -I/usr/include/tirpc   -c -o fact_server.o fact_server.c
cc -g -I/usr/include/tirpc    -o fact_server  fact_svc.o fact_server.o fact_xdr.o -lnsl -ltirpc -ltirpc */
```

Note:

We need to write only IDL file (fact.x), and edit the 3 template files:

1. **Makefile.fact**
2. **fact_server.c**
3. **fact_client.c**

I have used Bold and highlighted text to be updated in the template files in Yellow background in this document

Interface file: fact.x

```
struct intpair
{
        int a;
};

program FACT_PROG
{
version FACT_VERS
{
        int FACT(intpair) = 1;

} = 1;

} = 0x22222222;
```

To compile the program: Makefile.fact Template file

 # This is a template Makefile generated by rpcgen

 # Parameters

CLIENT = fact_client
SERVER = fact_server

SOURCES_CLNT.c =
SOURCES_CLNT.h =
SOURCES_SVC.c =
SOURCES_SVC.h =
SOURCES.x = fact.x

TARGETS_SVC.c = fact_svc.c fact_server.c fact_xdr.c
TARGETS_CLNT.c = fact_clnt.c fact_client.c fact_xdr.c
TARGETS = fact.h fact_xdr.c fact_clnt.c fact_svc.c fact_client.c fact_server.c

OBJECTS_CLNT = $(SOURCES_CLNT.c:%.c=%.o) $(TARGETS_CLNT.c:%.c=%.o)
OBJECTS_SVC = $(SOURCES_SVC.c:%.c=%.o) $(TARGETS_SVC.c:%.c=%.o)
 # Compiler flags

CFLAGS += -g
LDLIBS += -lnsl
RPCGENFLAGS =

 # Targets

all : $(CLIENT) $(SERVER)

$(TARGETS) : $(SOURCES.x)
rpcgen $(RPCGENFLAGS) $(SOURCES.x)

$(OBJECTS_CLNT) : $(SOURCES_CLNT.c) $(SOURCES_CLNT.h) $(TARGETS_CLNT.c)

$(OBJECTS_SVC) : $(SOURCES_SVC.c) $(SOURCES_SVC.h) $(TARGETS_SVC.c)

$(CLIENT) : $(OBJECTS_CLNT)
$(LINK.c) -o $(CLIENT) $(OBJECTS_CLNT) $(LDLIBS)

$(SERVER) : $(OBJECTS_SVC)
$(LINK.c) -o $(SERVER) $(OBJECTS_SVC) $(LDLIBS)

clean:
$(RM) core $(TARGETS) $(OBJECTS_CLNT) $(OBJECTS_SVC) $(CLIENT) $(SERVER)

To compile the program: Edited Makefile.fact file

This is a template Makefile generated by rpcgen

```
# Parameters

CLIENT = fact_client
SERVER = fact_server

SOURCES_CLNT.c =
SOURCES_CLNT.h =
SOURCES_SVC.c =
SOURCES_SVC.h =
SOURCES.x = fact.x

TARGETS_SVC.c = fact_svc.c fact_server.c fact_xdr.c
TARGETS_CLNT.c = fact_clnt.c fact_client.c fact_xdr.c
TARGETS = fact.h fact_xdr.c fact_clnt.c fact_svc.c fact_client.c fact_server.c

OBJECTS_CLNT = $(SOURCES_CLNT.c:%.c=%.o) $(TARGETS_CLNT.c:%.c=%.o)
OBJECTS_SVC = $(SOURCES_SVC.c:%.c=%.o) $(TARGETS_SVC.c:%.c=%.o)
# Compiler flags

CFLAGS += -g -I/usr/include/tirpc
LDLIBS += -lnsl -ltirpc
RPCGENFLAGS = -C
LIBS+= -ltirpc
QMAKE_CXXFLAGS+= -ltirpc

# Targets

all : $(CLIENT) $(SERVER)

$(TARGETS) : $(SOURCES.x)
	rpcgen $(RPCGENFLAGS) $(SOURCES.x)

$(OBJECTS_CLNT) : $(SOURCES_CLNT.c) $(SOURCES_CLNT.h) $(TARGETS_CLNT.c)

$(OBJECTS_SVC) : $(SOURCES_SVC.c) $(SOURCES_SVC.h) $(TARGETS_SVC.c)

$(CLIENT) : $(OBJECTS_CLNT)
	$(LINK.c) -o $(CLIENT) $(OBJECTS_CLNT) $(LDLIBS) $(LIBS)

$(SERVER) : $(OBJECTS_SVC)
	$(LINK.c) -o $(SERVER) $(OBJECTS_SVC) $(LDLIBS) $(LIBS)

 clean:
	 $(RM) core $(TARGETS) $(OBJECTS_CLNT) $(OBJECTS_SVC) $(CLIENT)
$(SERVER)
```

Server Code: fact_server .c Template file

```
/*
 * This is sample code generated by rpcgen.
```

```
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "fact.h"

int *
fact_1_svc(intpair *argp, struct svc_req *rqstp)
{
        static int  result;

        /*
         * insert server code here
         */

        return &result;
}
```

Server Code: fact_server.c Edited

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "fact.h"

int *
fact_1_svc(intpair *argp, struct svc_req *rqstp)
{
        static int  result,n,fact;

        int i;

        n=argp->a;

        // factorial logic

        fact = 1;
        printf("\n Received : n= %d \n",n);
        for (i=n;i>0;i--)
        {
                fact=fact * i;
        }

        result=fact;

        return &result;
}
```

Client Code: fact_client.c Template file

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "fact.h"


void
fact_prog_1(char *host)
{
        CLIENT *clnt;
        int  *result_1;
        intpair  fact_1_arg;

#ifndef         DEBUG
        clnt = clnt_create (host, FACT_PROG, FACT_VERS, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
#endif /* DEBUG */

        result_1 = fact_1(&fact_1_arg, clnt);
        if (result_1 == (int *) NULL) {
                clnt_perror (clnt, "call failed");
        }
#ifndef         DEBUG
        clnt_destroy (clnt);
#endif  /* DEBUG */
}


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

        if (argc < 2) {
                printf ("usage: %s server_host\n", argv[0]);
                exit (1);
        }
        host = argv[1];
        fact_prog_1 (host);
exit (0);
}
```

Client Code: Edited fact_client.c

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "fact.h"


void
fact_prog_1(char *host,int a)
{
        CLIENT *clnt;
        int  *result_1;
        intpair  fact_1_arg;

#ifndefDEBUG
        clnt = clnt_create (host, FACT_PROG, FACT_VERS, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
#endif  /* DEBUG */
fact_1_arg.a=a;

        result_1 = fact_1(&fact_1_arg, clnt);

        if (result_1 == (int *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        printf("\n Server returns=%d", *result_1);
#ifndefDEBUG
        clnt_destroy (clnt);
#endif   /* DEBUG */
}


int main (int argc, char *argv[])
{
char *host;
        int a,ch;
        if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
        }
        host = argv[1];
```

```
 do
{
system("clear");
printf("\nEnter a no:: ");
scanf("%d",&a);
fact_prog_1 (host,a);  // (Default statement: fact_prog_1 (host); )  edited with the ***do while code***

printf("\nTry again : (1/0) :: ");
scanf("%d",&ch);

}while(ch==1);

}
```

Server Running in one terminal:

```
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~$cd rpc
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~/rpc$ make -f Makefile.fact
make: Nothing to be done for 'all'.
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~/rpc$ ./fact_server

 Received : n= 1

 Received : n= 2

 Received : n= 3

 Received : n= 4

 Received : n= 5

 Received : n= 6

 Received : n= 7

 Received : n= 8

 Received : n= 9

 Received : n= 10
```

Client Running in another terminal:

```
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~/rpc$ ./fact_client localhost

Enter a no:: 9

 Server returns=362880
Try again : (1/0) :: 1
```

Enter a no:: 10

Server returns=3628800

Try again : (1/0) :: 0
(base) student@student-ASUS-EXPERTCENTER-D500MD-D500MD-IN:~/rpc$