**Assignment 4**

**Kanti**

**Gnana Deep**

**Usha**

**TCP:**

**Read Me**

A. To execute RPCGEN, we require this command: rpcgen rfour1.x

We can see the object files getting generated for both client and server.

Then, we have to seperate the server and the client files accordingly.

B. For Server,

1. To compile the .c files and link them together, we require this command: make -f Makefile.rfour1

2. Open the terminal, in which all the server files are stored.

3. To run the server, we should execute this command: ./rfour1d

C. For Client

1. To compile the .c files and link them together, we require this command: make all

2. Open the terminal, in which all the client files are stored.

3. To run the server, we should execute this command: ./rfour1 FILE\_NAME LOCATION

e.g. ./rfour1 testfile1 /home/gnanadeep/Pictures

4. Verify if the file is copied from the server to client location.

**rfour1.h**

/\*  
 \* Please do not edit this file.  
 \* It was generated using rpcgen.  
 \*/  
  
#ifndef \_RFOUR1\_H\_RPCGEN  
#define \_RFOUR1\_H\_RPCGEN  
  
#include <rpc/rpc.h>  
  
  
#ifdef \_\_cplusplus  
extern "C" {  
#endif  
  
#define MAXBUFLEN 10000  
  
#define FOUR1PROG 0x30090950  
#define FOUR1VERS 1  
  
#if defined(\_\_STDC\_\_) || defined(\_\_cplusplus)  
#define GETFILE 1  
extern char \*\* getfile\_1(char \*\*, CLIENT \*);  
extern char \*\* getfile\_1\_svc(char \*\*, struct svc\_req \*);  
extern int four1prog\_1\_freeresult (SVCXPRT \*, xdrproc\_t, caddr\_t);  
  
#else /\* K&R C \*/  
#define GETFILE 1  
extern char \*\* getfile\_1();  
extern char \*\* getfile\_1\_svc();  
extern int four1prog\_1\_freeresult ();  
#endif /\* K&R C \*/  
  
#ifdef \_\_cplusplus  
}  
#endif  
  
#endif /\* !\_RFOUR1\_H\_RPCGEN \*/

**rfour1.x**

const MAXBUFLEN=10000;

program FOUR1PROG

{

version FOUR1VERS

{

string GETFILE(string)=1;

}=1;

}=0x30090950;

**TCP Server Programs**

1. **rfour1\_svc.c**

/\*  
 \* Please do not edit this file.  
 \* It was generated using rpcgen.  
 \*/  
  
#include "rfour1.h"  
#include <stdio.h>  
#include <stdlib.h>  
#include <rpc/pmap\_clnt.h>  
#include <string.h>  
#include <memory.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
  
#ifndef SIG\_PF  
#define SIG\_PF void(\*)(int)  
#endif  
  
static void  
four1prog\_1(struct svc\_req \*rqstp, register SVCXPRT \*transp)  
{  
 union {  
 char \*getfile\_1\_arg;  
 } argument;  
 char \*result;  
 xdrproc\_t \_xdr\_argument, \_xdr\_result;  
 char \*(\*local)(char \*, struct svc\_req \*);  
  
 switch (rqstp->rq\_proc) {  
 case NULLPROC:  
 (void) svc\_sendreply (transp, (xdrproc\_t) xdr\_void, (char \*)NULL);  
 return;  
  
 case GETFILE:  
 \_xdr\_argument = (xdrproc\_t) xdr\_wrapstring;  
 \_xdr\_result = (xdrproc\_t) xdr\_wrapstring;  
 local = (char \*(\*)(char \*, struct svc\_req \*)) getfile\_1\_svc;  
 break;  
  
 default:  
 svcerr\_noproc (transp);  
 return;  
 }  
 memset ((char \*)&argument, 0, sizeof (argument));  
 if (!svc\_getargs (transp, (xdrproc\_t) \_xdr\_argument, (caddr\_t) &argument)) {  
 svcerr\_decode (transp);  
 return;  
 }  
 result = (\*local)((char \*)&argument, rqstp);  
 if (result != NULL && !svc\_sendreply(transp, (xdrproc\_t) \_xdr\_result, result)) {  
 svcerr\_systemerr (transp);  
 }  
 if (!svc\_freeargs (transp, (xdrproc\_t) \_xdr\_argument, (caddr\_t) &argument)) {  
 fprintf (stderr, "%s", "unable to free arguments");  
 exit (1);  
 }  
 return;  
}  
  
int  
main (int argc, char \*\*argv)  
{  
 register SVCXPRT \*transp;  
  
 pmap\_unset (FOUR1PROG, FOUR1VERS);  
  
 transp = svcudp\_create(RPC\_ANYSOCK);  
 if (transp == NULL) {  
 fprintf (stderr, "%s", "cannot create udp service.");  
 exit(1);  
 }  
 if (!svc\_register(transp, FOUR1PROG, FOUR1VERS, four1prog\_1, IPPROTO\_UDP)) {  
 fprintf (stderr, "%s", "unable to register (FOUR1PROG, FOUR1VERS, udp).");  
 exit(1);  
 }  
  
 transp = svctcp\_create(RPC\_ANYSOCK, 0, 0);  
 if (transp == NULL) {  
 fprintf (stderr, "%s", "cannot create tcp service.");  
 exit(1);  
 }  
 if (!svc\_register(transp, FOUR1PROG, FOUR1VERS, four1prog\_1, IPPROTO\_TCP)) {  
 fprintf (stderr, "%s", "unable to register (FOUR1PROG, FOUR1VERS, tcp).");  
 exit(1);  
 }  
  
 svc\_run ();  
 fprintf (stderr, "%s", "svc\_run returned");  
 exit (1);  
 /\* NOTREACHED \*/  
}

**2. rfour1\_srp**

#include <stdlib.h>  
#include <stdio.h>  
#include <ctype.h>  
#include <string.h>  
#include "rfour1.h"  
#include <rpc/rpc.h>  
#define MAXBUFLEN 10000  
  
char \*getfile(char \*filename);  
  
  
char \*getfile(char \*filename)  
{  
 FILE \*f1;  
 char \*buffer;  
 char ch;  
 while (1) {  
 //printf("\nWaiting for client\n");  
  
 printf("The file requested by the client is %s\n",filename);   
  
 f1=fopen(filename,"r");  
  
 buffer= malloc(MAXBUFLEN \* sizeof(char));  
 if (f1 != NULL) {  
 size\_t newLen = fread(buffer, sizeof(char), MAXBUFLEN+1, f1);  
 }  
 printf("File is opened and the content of the file is read\n\n");  
   
 //printf("%s",buffer);  
 fclose(f1);  
   
 return buffer;  
 }   
}  
  
**3. rfour1\_sif.c**

#include <rpc/rpc.h>  
  
#define RPC\_SVC  
#include "rfour1.h"  
  
static char \*retcode;  
  
extern char \*getfile(char \*);  
  
extern char \*\*  
getfile\_1\_svc(char \*\*w, struct svc\_req \*rqstp)  
{  
 retcode = getfile(\*(char\*\*)w);  
 return &retcode;  
}

**TCP Client Programs**

1. **rfour1.c**

#include <stdlib.h>  
#include <stdio.h>  
#include <ctype.h>  
#include <string.h>  
#include "rfour1.h"  
#define MAXBUFLEN 10000  
#define RMACHINE "localhost"  
CLIENT \*handle;  
  
char \*getfile(char \*filename);  
  
int main(int argc, char \*argv[])  
{  
 FILE \*f1;  
 char \*c1;  
  
 char \*filename=argv[1];  
 char \*location=argv[2];  
 handle=clnt\_create(RMACHINE,FOUR1PROG,FOUR1VERS,"tcp");  
 if(handle==0)  
 {  
 printf("could not contact\n");  
 exit(1);  
 }  
  
 char \*total=strcat(location,"/");  
 printf("Your have entered the filename: %s and location: %s\n",filename,location);  
 total=strcat(total,filename);  
 //printf("%s", total);  
 c1=getfile(filename);  
  
 f1=fopen(total,"w");  
 printf("%s\n",c1);  
 fputs(c1,f1);  
 printf("File is successfully copied to the specified location.\n");  
 fclose(f1);  
}

**2. rfour1\_clnt.c**

/\*  
 \* Please do not edit this file.  
 \* It was generated using rpcgen.  
 \*/  
  
#include <memory.h> /\* for memset \*/  
#include "rfour1.h"  
  
/\* Default timeout can be changed using clnt\_control() \*/  
static struct timeval TIMEOUT = { 25, 0 };  
  
char \*\*  
getfile\_1(char \*\*argp, CLIENT \*clnt)  
{  
 static char \*clnt\_res;  
  
 memset((char \*)&clnt\_res, 0, sizeof(clnt\_res));  
 if (clnt\_call (clnt, GETFILE,  
 (xdrproc\_t) xdr\_wrapstring, (caddr\_t) argp,  
 (xdrproc\_t) xdr\_wrapstring, (caddr\_t) &clnt\_res,  
 TIMEOUT) != RPC\_SUCCESS) {  
 return (NULL);  
 }  
 return (&clnt\_res);  
}

**3.rfour1\_cif.c**

#include <rpc/rpc.h>  
#include <stdio.h>  
#include "rfour1.h"  
  
extern CLIENT \*handle;  
static char \*\*ret;  
  
  
char \*getfile(char \*filename)  
{  
 char \*\*arg;  
 arg=&filename;  
 ret=getfile\_1(arg,handle);  
  
 return ret==0 ? 0 : \*ret;  
}

**Makefile**

# This is a template Makefile generated by rpcgen

# Parameters

CLIENT = rfour1

SERVER = rfour1d

SOURCES\_CLNT.c =

SOURCES\_CLNT.h =

SOURCES\_SVC.c =

SOURCES\_SVC.h =

SOURCES.x = rfour1.x

TARGETS\_SVC.c = rfour1\_svc.c rfour1\_srp.c

TARGETS\_CLNT.c = rfour1\_clnt.c rfour1.c

TARGETS = rfour1.h rfour1\_clnt.c rfour1\_svc.c rfour1.c rfour1\_srp.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)

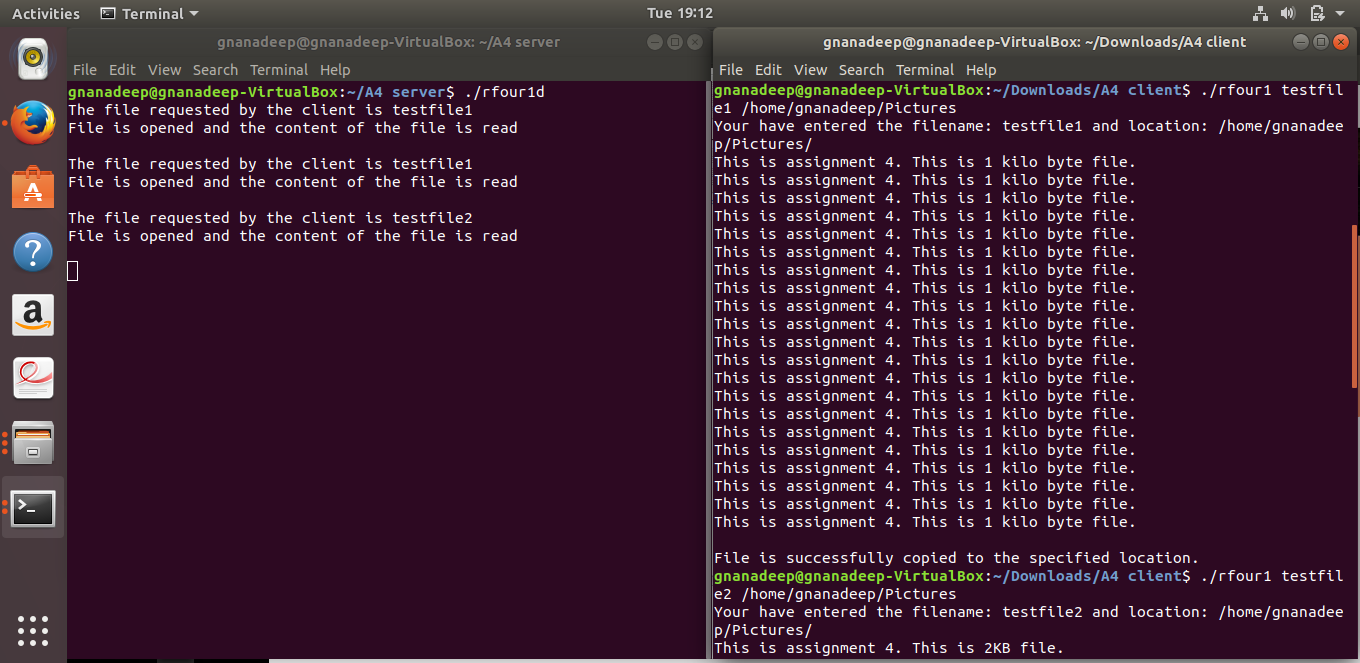
$(SERVER) : $(OBJECTS\_SVC)

$(LINK.c) -o $(SERVER) $(OBJECTS\_SVC) $(LDLIBS)

clean:

$(RM) core $(TARGETS) $(OBJECTS\_CLNT) $(OBJECTS\_SVC) $(CLIENT) $(SERVER)

**Output**



gnanadeep@gnanadeep-VirtualBox:~/Downloads/A4\_Client\_TCP-20171122T065406Z-001/A4\_Client\_TCP$ ./rfour1 testfile2 /home/gnanadeep/Pictures

Your have entered the filename: testfile2 and location: /home/gnanadeep/Pictures/

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

File is successfully copied to the specified location.

**UDP**

**Read Me**

A. To execute RPCGEN, we require this command: rpcgen rfour1.x

We can see the object files getting generated for both client and server.

Then, we have to seperate the server and the client files accordingly.

B. For Server,

1. To compile the .c files and link them together, we require this command: make -f Makefile.rfour1

2. Open the terminal, in which all the server files are stored.

3. To run the server, we should execute this command: ./rfour1d

C. For Client

1. To compile the .c files and link them together, we require this command: make all

2. Open the terminal, in which all the client files are stored.

3. To run the server, we should execute this command: ./rfour1 FILE\_NAME LOCATION

e.g. ./rfour1 testfile1 /home/gnanadeep/Pictures

4. Verify if the file is copied from the server to client location.

**rfour1.h**

/\*  
 \* Please do not edit this file.  
 \* It was generated using rpcgen.  
 \*/  
  
#ifndef \_RFOUR1\_H\_RPCGEN  
#define \_RFOUR1\_H\_RPCGEN  
  
#include <rpc/rpc.h>  
  
  
#ifdef \_\_cplusplus  
extern "C" {  
#endif  
  
#define MAXBUFLEN 10000  
  
#define FOUR1PROG 0x30090950  
#define FOUR1VERS 1  
  
#if defined(\_\_STDC\_\_) || defined(\_\_cplusplus)  
#define GETFILE 1  
extern char \*\* getfile\_1(char \*\*, CLIENT \*);  
extern char \*\* getfile\_1\_svc(char \*\*, struct svc\_req \*);  
extern int four1prog\_1\_freeresult (SVCXPRT \*, xdrproc\_t, caddr\_t);  
  
#else /\* K&R C \*/  
#define GETFILE 1  
extern char \*\* getfile\_1();  
extern char \*\* getfile\_1\_svc();  
extern int four1prog\_1\_freeresult ();  
#endif /\* K&R C \*/  
  
#ifdef \_\_cplusplus  
}  
#endif  
  
#endif /\* !\_RFOUR1\_H\_RPCGEN \*/

**rfour1.x**

const MAXBUFLEN=10000;

program FOUR1PROG

{

version FOUR1VERS

{

string GETFILE(string)=1;

}=1;

}=0x30090950;

**UDP Server Programs**

1. **rfour1\_svc.c**

/\*  
 \* Please do not edit this file.  
 \* It was generated using rpcgen.  
 \*/  
  
#include "rfour1.h"  
#include <stdio.h>  
#include <stdlib.h>  
#include <rpc/pmap\_clnt.h>  
#include <string.h>  
#include <memory.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
  
#ifndef SIG\_PF  
#define SIG\_PF void(\*)(int)  
#endif  
  
static void  
four1prog\_1(struct svc\_req \*rqstp, register SVCXPRT \*transp)  
{  
 union {  
 char \*getfile\_1\_arg;  
 } argument;  
 char \*result;  
 xdrproc\_t \_xdr\_argument, \_xdr\_result;  
 char \*(\*local)(char \*, struct svc\_req \*);  
  
 switch (rqstp->rq\_proc) {  
 case NULLPROC:  
 (void) svc\_sendreply (transp, (xdrproc\_t) xdr\_void, (char \*)NULL);  
 return;  
  
 case GETFILE:  
 \_xdr\_argument = (xdrproc\_t) xdr\_wrapstring;  
 \_xdr\_result = (xdrproc\_t) xdr\_wrapstring;  
 local = (char \*(\*)(char \*, struct svc\_req \*)) getfile\_1\_svc;  
 break;  
  
 default:  
 svcerr\_noproc (transp);  
 return;  
 }  
 memset ((char \*)&argument, 0, sizeof (argument));  
 if (!svc\_getargs (transp, (xdrproc\_t) \_xdr\_argument, (caddr\_t) &argument)) {  
 svcerr\_decode (transp);  
 return;  
 }  
 result = (\*local)((char \*)&argument, rqstp);  
 if (result != NULL && !svc\_sendreply(transp, (xdrproc\_t) \_xdr\_result, result)) {  
 svcerr\_systemerr (transp);  
 }  
 if (!svc\_freeargs (transp, (xdrproc\_t) \_xdr\_argument, (caddr\_t) &argument)) {  
 fprintf (stderr, "%s", "unable to free arguments");  
 exit (1);  
 }  
 return;  
}  
  
int  
main (int argc, char \*\*argv)  
{  
 register SVCXPRT \*transp;  
  
 pmap\_unset (FOUR1PROG, FOUR1VERS);  
  
 transp = svcudp\_create(RPC\_ANYSOCK);  
 if (transp == NULL) {  
 fprintf (stderr, "%s", "cannot create udp service.");  
 exit(1);  
 }  
 if (!svc\_register(transp, FOUR1PROG, FOUR1VERS, four1prog\_1, IPPROTO\_UDP)) {  
 fprintf (stderr, "%s", "unable to register (FOUR1PROG, FOUR1VERS, udp).");  
 exit(1);  
 }  
  
 transp = svctcp\_create(RPC\_ANYSOCK, 0, 0);  
 if (transp == NULL) {  
 fprintf (stderr, "%s", "cannot create tcp service.");  
 exit(1);  
 }  
 if (!svc\_register(transp, FOUR1PROG, FOUR1VERS, four1prog\_1, IPPROTO\_TCP)) {  
 fprintf (stderr, "%s", "unable to register (FOUR1PROG, FOUR1VERS, tcp).");  
 exit(1);  
 }  
  
 svc\_run ();  
 fprintf (stderr, "%s", "svc\_run returned");  
 exit (1);  
 /\* NOTREACHED \*/  
}

**2. rfour1\_srp**

#include <stdlib.h>  
#include <stdio.h>  
#include <ctype.h>  
#include <string.h>  
#include "rfour1.h"  
#include <rpc/rpc.h>  
#define MAXBUFLEN 10000  
  
char \*getfile(char \*filename);  
  
  
char \*getfile(char \*filename)  
{  
 FILE \*f1;  
 char \*buffer;  
 char ch;  
 while (1) {  
 //printf("\nWaiting for client\n");  
  
 printf("The file requested by the client is %s\n",filename);   
  
 f1=fopen(filename,"r");  
  
 buffer= malloc(MAXBUFLEN \* sizeof(char));  
 if (f1 != NULL) {  
 size\_t newLen = fread(buffer, sizeof(char), MAXBUFLEN+1, f1);  
 }  
 printf("File is opened and the content of the file is read\n\n");  
   
 //printf("%s",buffer);  
 fclose(f1);  
   
 return buffer;  
 }   
}  
  
**3. rfour1\_sif.c**

#include <rpc/rpc.h>  
  
#define RPC\_SVC  
#include "rfour1.h"  
  
static char \*retcode;  
  
extern char \*getfile(char \*);  
  
extern char \*\*  
getfile\_1\_svc(char \*\*w, struct svc\_req \*rqstp)  
{  
 retcode = getfile(\*(char\*\*)w);  
 return &retcode;  
}

**UDP Client Programs**

1. **rfour1.c**

#include <stdlib.h>  
#include <stdio.h>  
#include <ctype.h>  
#include <string.h>  
#include "rfour1.h"  
#define MAXBUFLEN 10000  
#define RMACHINE "localhost"  
CLIENT \*handle;  
  
char \*getfile(char \*filename);  
  
int main(int argc, char \*argv[])  
{  
 FILE \*f1;  
 char \*c1;  
  
 char \*filename=argv[1];  
 char \*location=argv[2];  
 handle=clnt\_create(RMACHINE,FOUR1PROG,FOUR1VERS,"udp");  
 if(handle==0)  
 {  
 printf("could not contact\n");  
 exit(1);  
 }  
  
 char \*total=strcat(location,"/");  
 printf("Your have entered the filename: %s and location: %s\n",filename,location);  
 total=strcat(total,filename);  
 //printf("%s", total);  
 c1=getfile(filename);  
  
 f1=fopen(total,"w");  
 printf("%s\n",c1);  
 fputs(c1,f1);  
 printf("File is successfully copied to the specified location.\n");  
 fclose(f1);  
}

**2. rfour1\_clnt.c**

/\*  
 \* Please do not edit this file.  
 \* It was generated using rpcgen.  
 \*/  
  
#include <memory.h> /\* for memset \*/  
#include "rfour1.h"  
  
/\* Default timeout can be changed using clnt\_control() \*/  
static struct timeval TIMEOUT = { 25, 0 };  
  
char \*\*  
getfile\_1(char \*\*argp, CLIENT \*clnt)  
{  
 static char \*clnt\_res;  
  
 memset((char \*)&clnt\_res, 0, sizeof(clnt\_res));  
 if (clnt\_call (clnt, GETFILE,  
 (xdrproc\_t) xdr\_wrapstring, (caddr\_t) argp,  
 (xdrproc\_t) xdr\_wrapstring, (caddr\_t) &clnt\_res,  
 TIMEOUT) != RPC\_SUCCESS) {  
 return (NULL);  
 }  
 return (&clnt\_res);  
}

**3.rfour1\_cif.c**

#include <rpc/rpc.h>  
#include <stdio.h>  
#include "rfour1.h"  
  
extern CLIENT \*handle;  
static char \*\*ret;  
  
  
char \*getfile(char \*filename)  
{  
 char \*\*arg;  
 arg=&filename;  
 ret=getfile\_1(arg,handle);  
  
 return ret==0 ? 0 : \*ret;  
}

**Makefile**

# This is a template Makefile generated by rpcgen

# Parameters

CLIENT = rfour1

SERVER = rfour1d

SOURCES\_CLNT.c =

SOURCES\_CLNT.h =

SOURCES\_SVC.c =

SOURCES\_SVC.h =

SOURCES.x = rfour1.x

TARGETS\_SVC.c = rfour1\_svc.c rfour1\_srp.c

TARGETS\_CLNT.c = rfour1\_clnt.c rfour1.c

TARGETS = rfour1.h rfour1\_clnt.c rfour1\_svc.c rfour1.c rfour1\_srp.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)

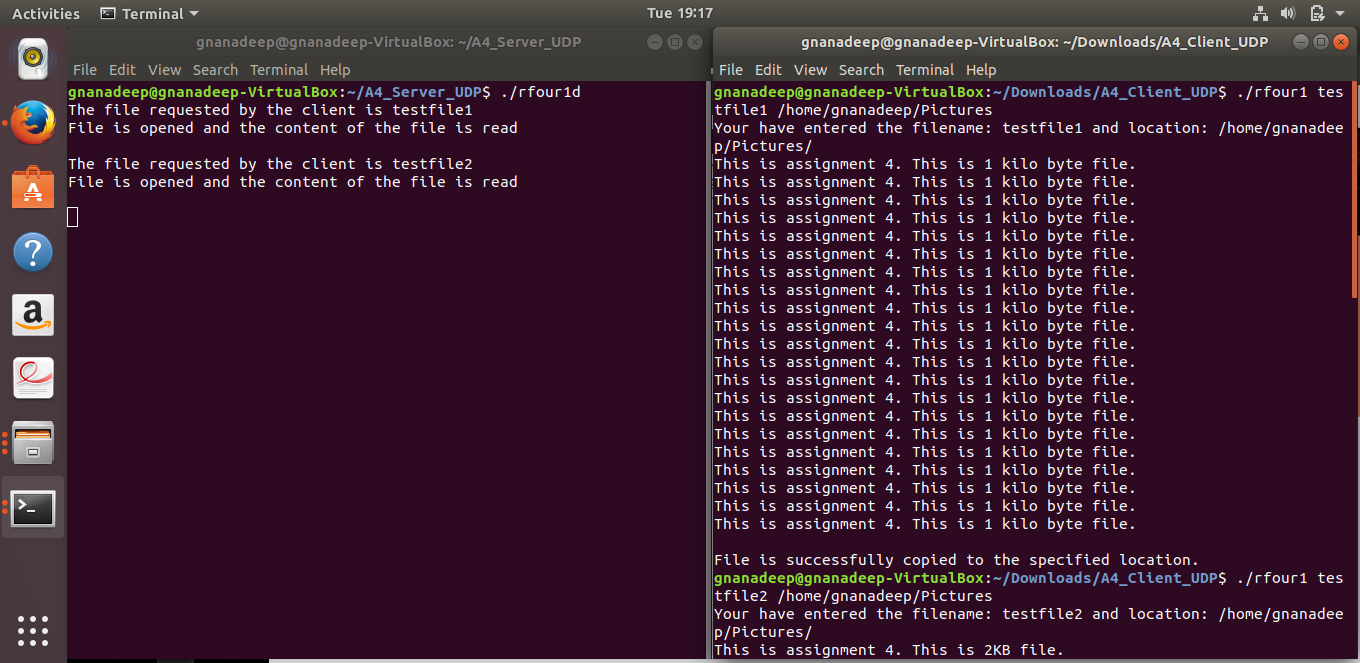
$(SERVER) : $(OBJECTS\_SVC)

$(LINK.c) -o $(SERVER) $(OBJECTS\_SVC) $(LDLIBS)

clean:

$(RM) core $(TARGETS) $(OBJECTS\_CLNT) $(OBJECTS\_SVC) $(CLIENT) $(SERVER)

**Output**

****

gnanadeep@gnanadeep-VirtualBox:~/Downloads/A4\_Client\_UDP-20171122T065406Z-001/A4\_Client\_UDP$ ./rfour1 testfile2 /home/gnanadeep/Pictures

Your have entered the filename: testfile2 and location: /home/gnanadeep/Pictures/

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

This is assignment 4. This is 2KB file.

File is successfully copied to the specified location.