\*\*NOTE : You can also find all files and programs on below link if you want download them and execute along with READ-ME.txt All Files are also pasted in this assignment. Please login with your SJSU account to view the link.

<https://drive.google.com/drive/folders/0BxDOhKoEJl-9MWs3dDY4TVFQMkE?usp=sharing>

Write a complete connection-oriented client – server program using RPC that performs the file server functionality using different file sizes: 800 and 1600 bytes.

Below are the program along with .x files,

Note : filestore.x file is same for both TCP and UDP.

**For TCP(connection oriented client-server),**

/\*--------------------------------------------Below is filestore.x file-----------------------------------------------\*/

const SERVER\_FILE\_PATH = "filestore/";

const CLIENT\_FILE\_PATH = "clientstore/";

typedef string filename<255>;

typedef opaque filechunk<1024>;

struct request {

filename name;

int pos;

};

program FILE\_STORE {

version FILE\_STORE\_VER\_1 {

int GETFILESIZE(filename name)=1;

filechunk GETFILEBYCHUNKS(request req)=2;

}=1;

}=0x23453333;

/\*-------------------------------------filestore\_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 "filestore.h"

#include <string.h>

#include <strings.h>

char\* getFullFilePath(char \*name);

void

file\_store\_1(char \*host,char \*name)

{

CLIENT \*clnt;

int \*result\_1;

filename getfilesize\_1\_arg=name;

filechunk \*result\_2;

request getfilebychunks\_1\_arg;

int pos=0, size=0;

getfilebychunks\_1\_arg.name = name; getfilebychunks\_1\_arg.pos=0;

#ifndef DEBUG

//clnt = clnt\_create (host, FILE\_STORE, FILE\_STORE\_VER\_1, "udp");

clnt = clnt\_create (host, FILE\_STORE, FILE\_STORE\_VER\_1, "tcp");

if (clnt == NULL) {

clnt\_pcreateerror (host);

exit (1);

}

#endif /\* DEBUG \*/

result\_1 = getfilesize\_1(&getfilesize\_1\_arg, clnt); printf("%d",\*result\_1);

size = \*result\_1;

if (result\_1 == (int \*) NULL) {

clnt\_perror (clnt, "call failed");

}

char \*fullpath = getFullFilePath(name);

FILE \*fp;

if(size >0 ) {

fp = fopen(fullpath,"w+");

}

while(size > 0) {

result\_2 = getfilebychunks\_1(&getfilebychunks\_1\_arg, clnt);

if (result\_2 == (filechunk \*) NULL) {

clnt\_perror (clnt, "call failed");

}

//printf("%s",result\_2->filechunk\_val);

fputs(result\_2->filechunk\_val, fp);

int bytesread=strlen(result\_2->filechunk\_val);

size-=bytesread;

pos+=bytesread;

getfilebychunks\_1\_arg.pos=pos;

}

fclose(fp);

#ifndef DEBUG

clnt\_destroy (clnt);

#endif /\* DEBUG \*/

}

char\* getFullFilePath(char \*name) {

char \*full\_path = malloc(strlen(CLIENT\_FILE\_PATH)+1);

memset(full\_path,'\0',strlen(CLIENT\_FILE\_PATH)+1);

strcpy(full\_path,CLIENT\_FILE\_PATH);

full\_path= realloc(full\_path,strlen(full\_path)+strlen(name));

strcat(full\_path,name); printf("%s",full\_path);

return full\_path;

}

int

main (int argc, char \*argv[])

{

char \*host, \*filename;

if (argc < 3) {

printf ("usage: %s server\_host file\_name\n", argv[0]);

exit (1);

}

host = argv[1];

filename = argv[2];

file\_store\_1 (host,filename);

exit (0);

}

/\*-------------------------------------filestore\_clnt.c---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#include <memory.h> /\* for memset \*/

#include "filestore.h"

/\* Default timeout can be changed using clnt\_control() \*/

static struct timeval TIMEOUT = { 25, 0 };

int \*

getfilesize\_1(filename \*argp, CLIENT \*clnt)

{

static int clnt\_res;

memset((char \*)&clnt\_res, 0, sizeof(clnt\_res));

if (clnt\_call (clnt, GETFILESIZE,

(xdrproc\_t) xdr\_filename, (caddr\_t) argp,

(xdrproc\_t) xdr\_int, (caddr\_t) &clnt\_res,

TIMEOUT) != RPC\_SUCCESS) {

return (NULL);

}

return (&clnt\_res);

}

filechunk \*

getfilebychunks\_1(request \*argp, CLIENT \*clnt)

{

static filechunk clnt\_res;

memset((char \*)&clnt\_res, 0, sizeof(clnt\_res));

if (clnt\_call (clnt, GETFILEBYCHUNKS,

(xdrproc\_t) xdr\_request, (caddr\_t) argp,

(xdrproc\_t) xdr\_filechunk, (caddr\_t) &clnt\_res,

TIMEOUT) != RPC\_SUCCESS) {

return (NULL);

}

return (&clnt\_res);

}

/\*-------------------------------------filestore\_server.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 "filestore.h"

#include <string.h>

#include <strings.h>

int getFileSize(char\* name);

char\* getFullFilePath(char \*name);

char\* getFileByChunks(char\* name, int pos);

int \*

getfilesize\_1\_svc(filename \*argp, struct svc\_req \*rqstp)

{

static int result=0;

/\*

\* insert server code here

\*/

if(argp) {

printf("%s",\*argp);

result= getFileSize(\*argp);

}

printf("%d",result);

return &result;

}

filechunk \*

getfilebychunks\_1\_svc(request \*argp, struct svc\_req \*rqstp)

{

static filechunk result;

/\*

\* insert server code here

\*/

result.filechunk\_val = getFileByChunks(argp->name,argp->pos);

result.filechunk\_len = strlen(result.filechunk\_val);

return &result;

}

/\*These are my internal function definitions\*/

int getFileSize(char\* name) {

int size=0;

if(name != NULL) {

char\* full\_path = getFullFilePath(name);

FILE \*fp = fopen(full\_path,"r");

fseek(fp,0,SEEK\_END);

size=ftell(fp);

rewind(fp);

fclose(fp);

free(full\_path);

}

return size;

}

char\* getFullFilePath(char \*name) {

char \*full\_path = malloc(strlen(SERVER\_FILE\_PATH)+1);

memset(full\_path,'\0',strlen(SERVER\_FILE\_PATH)+1);

strcpy(full\_path,SERVER\_FILE\_PATH);

full\_path= realloc(full\_path,strlen(full\_path)+strlen(name));

strcat(full\_path,name); printf("%s",full\_path);

return full\_path;

}

char\* getFileByChunks(char\* name, int pos){

char \*chunk= malloc(1024); memset(chunk,'\0',1024);

memset(chunk,'\0',1024);

char\* full\_path = getFullFilePath(name);

FILE \*fp = fopen(full\_path,"r");

if(fp==NULL){

return NULL;

}

fseek(fp,pos,SEEK\_SET);

fgets(chunk,1024,fp);

fclose(fp);

return chunk;

}

/\*-------------------------------------filestore\_svc.c---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#include "filestore.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

file\_store\_1(struct svc\_req \*rqstp, register SVCXPRT \*transp)

{

union {

filename getfilesize\_1\_arg;

request getfilebychunks\_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 GETFILESIZE:

\_xdr\_argument = (xdrproc\_t) xdr\_filename;

\_xdr\_result = (xdrproc\_t) xdr\_int;

local = (char \*(\*)(char \*, struct svc\_req \*)) getfilesize\_1\_svc;

break;

case GETFILEBYCHUNKS:

\_xdr\_argument = (xdrproc\_t) xdr\_request;

\_xdr\_result = (xdrproc\_t) xdr\_filechunk;

local = (char \*(\*)(char \*, struct svc\_req \*)) getfilebychunks\_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 (FILE\_STORE, FILE\_STORE\_VER\_1);

/\*transp = svcudp\_create(RPC\_ANYSOCK);

if (transp == NULL) {

fprintf (stderr, "%s", "cannot create udp service.");

exit(1);

}

if (!svc\_register(transp, FILE\_STORE, FILE\_STORE\_VER\_1, file\_store\_1, IPPROTO\_UDP)) {

fprintf (stderr, "%s", "unable to register (FILE\_STORE, FILE\_STORE\_VER\_1, 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, FILE\_STORE, FILE\_STORE\_VER\_1, file\_store\_1, IPPROTO\_TCP)) {

fprintf (stderr, "%s", "unable to register (FILE\_STORE, FILE\_STORE\_VER\_1, tcp).");

exit(1);

}

svc\_run ();

fprintf (stderr, "%s", "svc\_run returned");

exit (1);

/\* NOTREACHED \*/

}

/\*-------------------------------------filestore\_xdr.c---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#include "filestore.h"

bool\_t

xdr\_filename (XDR \*xdrs, filename \*objp)

{

register int32\_t \*buf;

if (!xdr\_string (xdrs, objp, 255))

return FALSE;

return TRUE;

}

bool\_t

xdr\_filechunk (XDR \*xdrs, filechunk \*objp)

{

register int32\_t \*buf;

if (!xdr\_bytes (xdrs, (char \*\*)&objp->filechunk\_val, (u\_int \*) &objp->filechunk\_len, 1024))

return FALSE;

return TRUE;

}

bool\_t

xdr\_request (XDR \*xdrs, request \*objp)

{

register int32\_t \*buf;

if (!xdr\_filename (xdrs, &objp->name))

return FALSE;

if (!xdr\_int (xdrs, &objp->pos))

return FALSE;

return TRUE;

}

/\*-------------------------------------filestore.h---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#ifndef \_FILESTORE\_H\_RPCGEN

#define \_FILESTORE\_H\_RPCGEN

#include <rpc/rpc.h>

#ifdef \_\_cplusplus

extern "C" {

#endif

#define SERVER\_FILE\_PATH "filestore/"

#define CLIENT\_FILE\_PATH "clientstore/"

typedef char \*filename;

typedef struct {

u\_int filechunk\_len;

char \*filechunk\_val;

} filechunk;

struct request {

filename name;

int pos;

};

typedef struct request request;

#define FILE\_STORE 0x23453333

#define FILE\_STORE\_VER\_1 1

#if defined(\_\_STDC\_\_) || defined(\_\_cplusplus)

#define GETFILESIZE 1

extern int \* getfilesize\_1(filename \*, CLIENT \*);

extern int \* getfilesize\_1\_svc(filename \*, struct svc\_req \*);

#define GETFILEBYCHUNKS 2

extern filechunk \* getfilebychunks\_1(request \*, CLIENT \*);

extern filechunk \* getfilebychunks\_1\_svc(request \*, struct svc\_req \*);

extern int file\_store\_1\_freeresult (SVCXPRT \*, xdrproc\_t, caddr\_t);

#else /\* K&R C \*/

#define GETFILESIZE 1

extern int \* getfilesize\_1();

extern int \* getfilesize\_1\_svc();

#define GETFILEBYCHUNKS 2

extern filechunk \* getfilebychunks\_1();

extern filechunk \* getfilebychunks\_1\_svc();

extern int file\_store\_1\_freeresult ();

#endif /\* K&R C \*/

/\* the xdr functions \*/

#if defined(\_\_STDC\_\_) || defined(\_\_cplusplus)

extern bool\_t xdr\_filename (XDR \*, filename\*);

extern bool\_t xdr\_filechunk (XDR \*, filechunk\*);

extern bool\_t xdr\_request (XDR \*, request\*);

#else /\* K&R C \*/

extern bool\_t xdr\_filename ();

extern bool\_t xdr\_filechunk ();

extern bool\_t xdr\_request ();

#endif /\* K&R C \*/

#ifdef \_\_cplusplus

}

#endif

#endif /\* !\_FILESTORE\_H\_RPCGEN \*/

**2. For UDP(connection oriented client-server),**

/\*--------------------------------------------Below is filestore.x file-----------------------------------------------\*/

const SERVER\_FILE\_PATH = "filestore/";

const CLIENT\_FILE\_PATH = "clientstore/";

typedef string filename<255>;

typedef opaque filechunk<1024>;

struct request {

filename name;

int pos;

};

program FILE\_STORE {

version FILE\_STORE\_VER\_1 {

int GETFILESIZE(filename name)=1;

filechunk GETFILEBYCHUNKS(request req)=2;

}=1;

}=0x23453333;

/\*-------------------------------------filestore\_xdr.c---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#include "filestore.h"

bool\_t

xdr\_filename (XDR \*xdrs, filename \*objp)

{

register int32\_t \*buf;

if (!xdr\_string (xdrs, objp, 255))

return FALSE;

return TRUE;

}

bool\_t

xdr\_filechunk (XDR \*xdrs, filechunk \*objp)

{

register int32\_t \*buf;

if (!xdr\_bytes (xdrs, (char \*\*)&objp->filechunk\_val, (u\_int \*) &objp->filechunk\_len, 1024))

return FALSE;

return TRUE;

}

bool\_t

xdr\_request (XDR \*xdrs, request \*objp)

{

register int32\_t \*buf;

if (!xdr\_filename (xdrs, &objp->name))

return FALSE;

if (!xdr\_int (xdrs, &objp->pos))

return FALSE;

return TRUE;

}

/\*-------------------------------------filestore.h---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#ifndef \_FILESTORE\_H\_RPCGEN

#define \_FILESTORE\_H\_RPCGEN

#include <rpc/rpc.h>

#ifdef \_\_cplusplus

extern "C" {

#endif

#define SERVER\_FILE\_PATH "filestore/"

#define CLIENT\_FILE\_PATH "clientstore/"

typedef char \*filename;

typedef struct {

u\_int filechunk\_len;

char \*filechunk\_val;

} filechunk;

struct request {

filename name;

int pos;

};

typedef struct request request;

#define FILE\_STORE 0x23453333

#define FILE\_STORE\_VER\_1 1

#if defined(\_\_STDC\_\_) || defined(\_\_cplusplus)

#define GETFILESIZE 1

extern int \* getfilesize\_1(filename \*, CLIENT \*);

extern int \* getfilesize\_1\_svc(filename \*, struct svc\_req \*);

#define GETFILEBYCHUNKS 2

extern filechunk \* getfilebychunks\_1(request \*, CLIENT \*);

extern filechunk \* getfilebychunks\_1\_svc(request \*, struct svc\_req \*);

extern int file\_store\_1\_freeresult (SVCXPRT \*, xdrproc\_t, caddr\_t);

#else /\* K&R C \*/

#define GETFILESIZE 1

extern int \* getfilesize\_1();

extern int \* getfilesize\_1\_svc();

#define GETFILEBYCHUNKS 2

extern filechunk \* getfilebychunks\_1();

extern filechunk \* getfilebychunks\_1\_svc();

extern int file\_store\_1\_freeresult ();

#endif /\* K&R C \*/

/\* the xdr functions \*/

#if defined(\_\_STDC\_\_) || defined(\_\_cplusplus)

extern bool\_t xdr\_filename (XDR \*, filename\*);

extern bool\_t xdr\_filechunk (XDR \*, filechunk\*);

extern bool\_t xdr\_request (XDR \*, request\*);

#else /\* K&R C \*/

extern bool\_t xdr\_filename ();

extern bool\_t xdr\_filechunk ();

extern bool\_t xdr\_request ();

#endif /\* K&R C \*/

#ifdef \_\_cplusplus

}

#endif

#endif /\* !\_FILESTORE\_H\_RPCGEN \*/

/\*-------------------------------------filestore\_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 "filestore.h"

#include <string.h>

#include <strings.h>

char\* getFullFilePath(char \*name);

void

file\_store\_1(char \*host,char \*name)

{

CLIENT \*clnt;

int \*result\_1;

filename getfilesize\_1\_arg=name;

filechunk \*result\_2;

request getfilebychunks\_1\_arg;

int pos=0, size=0;

getfilebychunks\_1\_arg.name = name; getfilebychunks\_1\_arg.pos=0;

#ifndef DEBUG

clnt = clnt\_create (host, FILE\_STORE, FILE\_STORE\_VER\_1, "udp");

//clnt = clnt\_create (host, FILE\_STORE, FILE\_STORE\_VER\_1, "tcp");

if (clnt == NULL) {

clnt\_pcreateerror (host);

exit (1);

}

#endif /\* DEBUG \*/

result\_1 = getfilesize\_1(&getfilesize\_1\_arg, clnt); printf("%d",\*result\_1);

size = \*result\_1;

if (result\_1 == (int \*) NULL) {

clnt\_perror (clnt, "call failed");

}

char \*fullpath = getFullFilePath(name);

FILE \*fp;

if(size >0 ) {

fp = fopen(fullpath,"w+");

}

while(size > 0) {

result\_2 = getfilebychunks\_1(&getfilebychunks\_1\_arg, clnt);

if (result\_2 == (filechunk \*) NULL) {

clnt\_perror (clnt, "call failed");

}

//printf("%s",result\_2->filechunk\_val);

fputs(result\_2->filechunk\_val, fp);

int bytesread=strlen(result\_2->filechunk\_val);

size-=bytesread;

pos+=bytesread;

getfilebychunks\_1\_arg.pos=pos;

}

fclose(fp);

#ifndef DEBUG

clnt\_destroy (clnt);

#endif /\* DEBUG \*/

}

char\* getFullFilePath(char \*name) {

char \*full\_path = malloc(strlen(CLIENT\_FILE\_PATH)+1);

memset(full\_path,'\0',strlen(CLIENT\_FILE\_PATH)+1);

strcpy(full\_path,CLIENT\_FILE\_PATH);

full\_path= realloc(full\_path,strlen(full\_path)+strlen(name));

strcat(full\_path,name); printf("%s",full\_path);

return full\_path;

}

int

main (int argc, char \*argv[])

{

char \*host, \*filename;

if (argc < 3) {

printf ("usage: %s server\_host file\_name\n", argv[0]);

exit (1);

}

host = argv[1];

filename = argv[2];

file\_store\_1 (host,filename);

exit (0);

}

/\*-------------------------------------filestore\_clnt.c---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#include <memory.h> /\* for memset \*/

#include "filestore.h"

/\* Default timeout can be changed using clnt\_control() \*/

static struct timeval TIMEOUT = { 25, 0 };

int \*

getfilesize\_1(filename \*argp, CLIENT \*clnt)

{

static int clnt\_res;

memset((char \*)&clnt\_res, 0, sizeof(clnt\_res));

if (clnt\_call (clnt, GETFILESIZE,

(xdrproc\_t) xdr\_filename, (caddr\_t) argp,

(xdrproc\_t) xdr\_int, (caddr\_t) &clnt\_res,

TIMEOUT) != RPC\_SUCCESS) {

return (NULL);

}

return (&clnt\_res);

}

filechunk \*

getfilebychunks\_1(request \*argp, CLIENT \*clnt)

{

static filechunk clnt\_res;

memset((char \*)&clnt\_res, 0, sizeof(clnt\_res));

if (clnt\_call (clnt, GETFILEBYCHUNKS,

(xdrproc\_t) xdr\_request, (caddr\_t) argp,

(xdrproc\_t) xdr\_filechunk, (caddr\_t) &clnt\_res,

TIMEOUT) != RPC\_SUCCESS) {

return (NULL);

}

return (&clnt\_res);

}

/\*-------------------------------------filestore\_server.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 "filestore.h"

#include <string.h>

#include <strings.h>

int getFileSize(char\* name);

char\* getFullFilePath(char \*name);

char\* getFileByChunks(char\* name, int pos);

int \*

getfilesize\_1\_svc(filename \*argp, struct svc\_req \*rqstp)

{

static int result=0;

/\*

\* insert server code here

\*/

if(argp) {

printf("%s",\*argp);

result= getFileSize(\*argp);

}

printf("%d",result);

return &result;

}

filechunk \*

getfilebychunks\_1\_svc(request \*argp, struct svc\_req \*rqstp)

{

static filechunk result;

/\*

\* insert server code here

\*/

result.filechunk\_val = getFileByChunks(argp->name,argp->pos);

result.filechunk\_len = strlen(result.filechunk\_val);

return &result;

}

/\*These are my internal function definitions\*/

int getFileSize(char\* name) {

int size=0;

if(name != NULL) {

char\* full\_path = getFullFilePath(name);

FILE \*fp = fopen(full\_path,"r");

fseek(fp,0,SEEK\_END);

size=ftell(fp);

rewind(fp);

fclose(fp);

free(full\_path);

}

return size;

}

char\* getFullFilePath(char \*name) {

char \*full\_path = malloc(strlen(SERVER\_FILE\_PATH)+1);

memset(full\_path,'\0',strlen(SERVER\_FILE\_PATH)+1);

strcpy(full\_path,SERVER\_FILE\_PATH);

full\_path= realloc(full\_path,strlen(full\_path)+strlen(name));

strcat(full\_path,name); printf("%s",full\_path);

return full\_path;

}

char\* getFileByChunks(char\* name, int pos){

char \*chunk= malloc(1024); memset(chunk,'\0',1024);

memset(chunk,'\0',1024);

char\* full\_path = getFullFilePath(name);

FILE \*fp = fopen(full\_path,"r");

if(fp==NULL){

return NULL;

}

fseek(fp,pos,SEEK\_SET);

fgets(chunk,1024,fp);

fclose(fp);

return chunk;

}

/\*-------------------------------------filestore\_svc.c---------------------------------\*/

/\*

\* Please do not edit this file.

\* It was generated using rpcgen.

\*/

#include "filestore.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

file\_store\_1(struct svc\_req \*rqstp, register SVCXPRT \*transp)

{

union {

filename getfilesize\_1\_arg;

request getfilebychunks\_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 GETFILESIZE:

\_xdr\_argument = (xdrproc\_t) xdr\_filename;

\_xdr\_result = (xdrproc\_t) xdr\_int;

local = (char \*(\*)(char \*, struct svc\_req \*)) getfilesize\_1\_svc;

break;

case GETFILEBYCHUNKS:

\_xdr\_argument = (xdrproc\_t) xdr\_request;

\_xdr\_result = (xdrproc\_t) xdr\_filechunk;

local = (char \*(\*)(char \*, struct svc\_req \*)) getfilebychunks\_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 (FILE\_STORE, FILE\_STORE\_VER\_1);

transp = svcudp\_create(RPC\_ANYSOCK);

if (transp == NULL) {

fprintf (stderr, "%s", "cannot create udp service.");

exit(1);

}

if (!svc\_register(transp, FILE\_STORE, FILE\_STORE\_VER\_1, file\_store\_1, IPPROTO\_UDP)) {

fprintf (stderr, "%s", "unable to register (FILE\_STORE, FILE\_STORE\_VER\_1, 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, FILE\_STORE, FILE\_STORE\_VER\_1, file\_store\_1, IPPROTO\_TCP)) {

fprintf (stderr, "%s", "unable to register (FILE\_STORE, FILE\_STORE\_VER\_1, tcp).");

exit(1);

}\*/

svc\_run ();

fprintf (stderr, "%s", "svc\_run returned");

exit (1);

/\* NOTREACHED \*/

}

/\*--------------------------------------------Below is Makefile.filestore-------------------------------------------\*/

# This is a template Makefile generated by rpcgen

# Parameters

CLIENT = filestore\_client

SERVER = filestore\_server

SOURCES\_CLNT.c =

SOURCES\_CLNT.h =

SOURCES\_SVC.c =

SOURCES\_SVC.h =

SOURCES.x = filestore.x

TARGETS\_SVC.c = filestore\_svc.c filestore\_server.c filestore\_xdr.c

TARGETS\_CLNT.c = filestore\_clnt.c filestore\_client.c filestore\_xdr.c

TARGETS = filestore.h filestore\_xdr.c filestore\_clnt.c filestore\_svc.c filestore\_client.c filestore\_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)

/\*--------------------------------------------READ-ME.txt-------------------------------------------\*/

To install and configure RPCGEN :

1. First check if rpc in already installed or not using command,

rpcinfo

2.If not install RPCGEN using

sudo apt-get install rpcbind

3.Write .x file to generate source code for RPC

Please refer to filestore.x file

4.Compile it using following command,

rpcgen -a -C filestore.x (Note that -C is capital)

Above command will generate filestore\_\*.c files for server and client along with make file to compile

\*\*Note : If you run this command our .c files will get overwritten and will loose logic placed in it for file transfer in step 5.

5.Edit filestore\_server.c and filestore\_client.c for putting your logic in code

\*\*Note : This has been already done in \*.c files we provided

6. Compile using make -f Makefile.filestore

\*\*Note : Since we have already provided \*.c files its better to run this step directly.

commands to run the program :

1. create 2 folders for storing files

a. At server side make ./filestore folder and place files in it (we have file1.txt and file2.txt there)

b. At client side make ./clientstore folder

2. run server using ./filestore\_server

3. run client using ./filestore\_client localhost file1.txt

4. check for file in clientstore dir at client side

/\*-------------------------------------./filestore/file1.txt---------------------------------\*/

Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

/\*-------------------------------------./filestore/file2.txt---------------------------------\*/

Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

\*\*NOTE : You can also find all files and programs on below link if you want download them and execute along with READ-ME.txt

<https://drive.google.com/drive/folders/0BxDOhKoEJl-9MWs3dDY4TVFQMkE?usp=sharing>