Name: Ken Lin

Github repository of my code : https://github.com/DehNutCase/CSE-461/tree/master/lab5

I believe all parts are completed successfully.

However, I don't know how many points they are worth (the lab 5 page doesn't say), so I assumed 20 points.

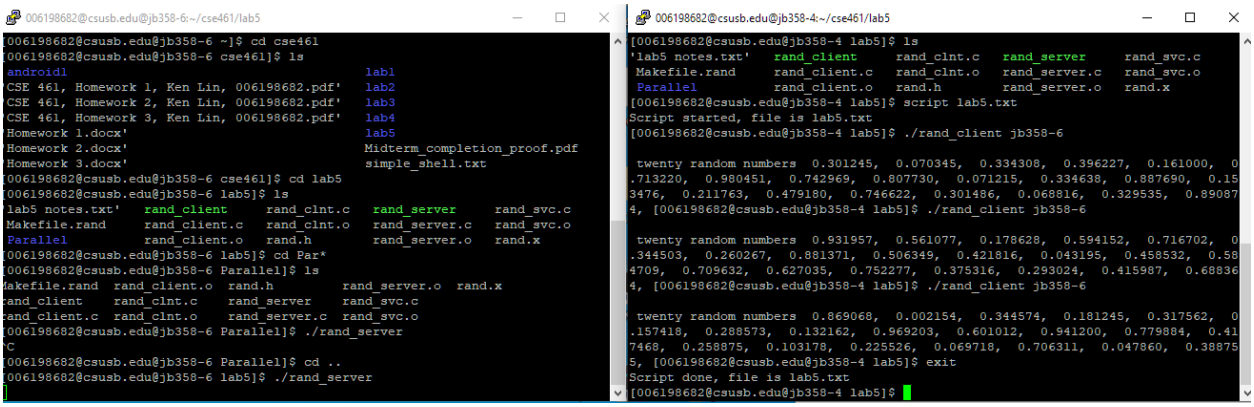
19 points.

Report:

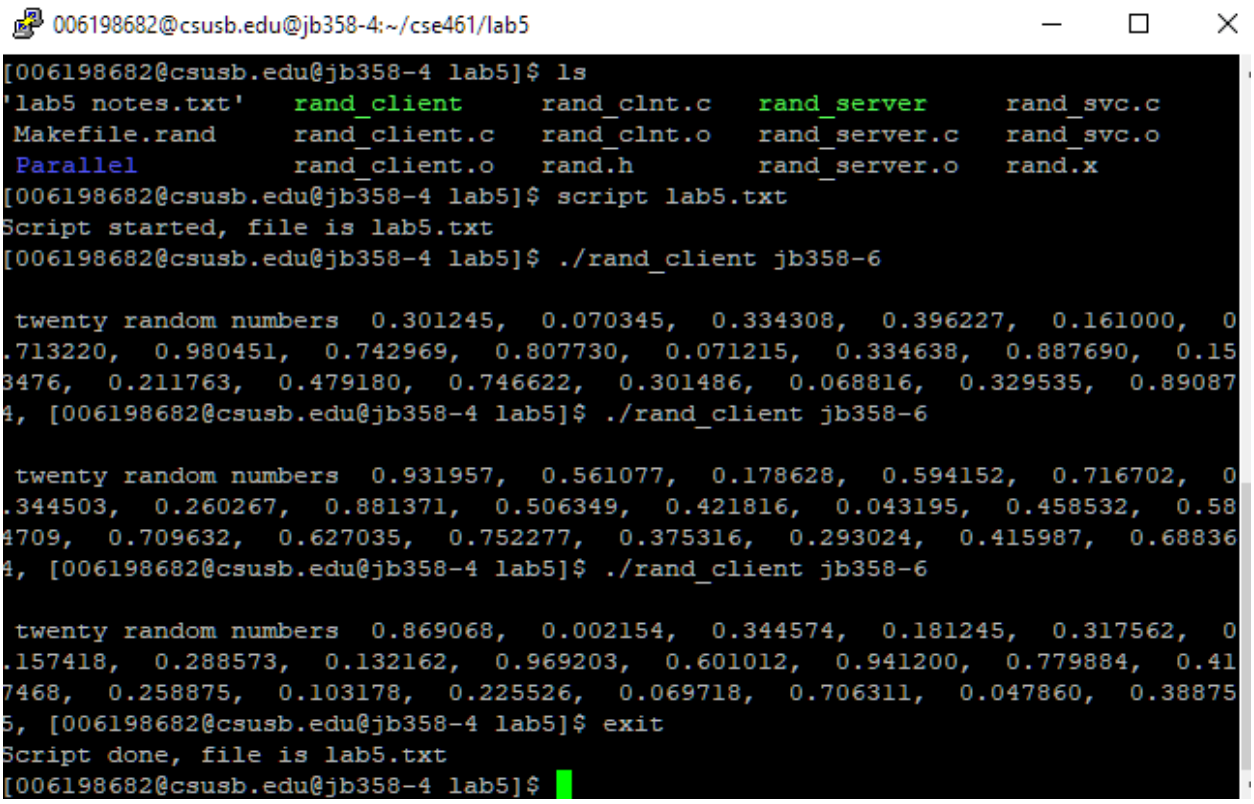
Parts 1 to 8: (10 points assumed)

Remote Procedure Call, Random Number Generator.

Screenshots:



Screenshot of client and server running simultaneously.



Client by itself.

Script File:

Lab5.txt (Script of me running the client program.)

Script started on 2020-06-02 23:34:46-07:00 [TERM="xterm" TTY="/dev/pts/0" COLUMNS="80" LINES="24"]
]0;006198682@csusb.edu@jb358-4:~/cse461/lab5 [006198682@csusb.edu@jb358-4 lab5]$ ./rand_client jb [K [K [K [Kt jb358-6
[006198682@csusb.edu@jb358-4 lab5]$ ./rand_client jb [K [K [K [Kt jb358-6

twenty random numbers 0.301245, 0.070345, 0.334308, 0.396227, 0.161000, 0.713220, 0.980451, 0.742969, 0.807730, 0.071215, 0.334638, 0.887690, 0.153476, 0.211763, 0.479180, 0.746622, 0.301486, 0.068816, 0.329535, 0.890874,
]0;006198682@csusb.edu@jb358-4:~/cse461/lab5 [006198682@csusb.edu@jb358-4 lab5]$ ./rand_client jb358-6
[006198682@csusb.edu@jb358-4 lab5]$ ./rand_client jb358-6

twenty random numbers 0.931957, 0.561077, 0.178628, 0.594152, 0.716702, 0.344503, 0.260267, 0.881371, 0.506349, 0.421816, 0.043195, 0.458532, 0.584709, 0.709632, 0.627035, 0.752277, 0.375316, 0.293024, 0.415987, 0.688364,

```
   ]0;006198682@csusb.edu@jb358-4:~/cse461/lab5   [006198682@csusb.edu@jb358-4
lab5]$ ./rand_client jb358-6
[006198682@csusb.edu@jb358-4 lab5]$ ./rand_client jb358-6

 twenty random numbers  0.869068,  0.002154,  0.344574,  0.181245,  0.317562,  0.157418,
0.288573,  0.132162,  0.969203,  0.601012,  0.941200,  0.779884,  0.417468,  0.258875,
0.103178,  0.225526,  0.069718,  0.706311,  0.047860,  0.388755,
   ]0;006198682@csusb.edu@jb358-4:~/cse461/lab5   [006198682@csusb.edu@jb358-4
lab5]$ exit
[006198682@csusb.edu@jb358-4 lab5]$ exit

Script done on 2020-06-02 23:35:15-07:00 [COMMAND_EXIT_CODE="0"]
```

Source Code:

Rand.x:

(Identical for this and part 9)

```
/* rand.x */

program RAND_PROG {
  version RAND_VERS {
    void INITIALIZE_RANDOM ( long ) = 1;        /* service #1 */
    double GET_NEXT_RANDOM ( void ) = 2;        /* service #2 */
  } = 1;
} = 6198682;         /* program # */
```

Rand_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 "rand.h"


double
rand_prog_1(char *host)
{
        CLIENT *clnt;
        void  *result_1;
```

```
        long  initialize_random_1_arg;
        double  *result_2;
        char *get_next_random_1_arg;

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

        result_1 = initialize_random_1(&initialize_random_1_arg, clnt);
        if (result_1 == (void *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        result_2 = get_next_random_1((void*)&get_next_random_1_arg, clnt);
        if (result_2 == (double *) NULL) {
                clnt_perror (clnt, "call failed");
        }
#ifndef DEBUG
        clnt_destroy (clnt);
#endif   /* DEBUG */

  return *result_2;
}


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

        if (argc < 2) {
                printf ("usage: %s server_host\n", argv[0]);
                exit (1);
        }
        host = argv[1];
        rand_prog_1 (host);

  double x;
     int i;
     printf("\n twenty random numbers ");
```

```
    for ( i = 0; i < 20; ++i ){
        x = rand_prog_1 (host);
        printf(" %f, ", x );
    }
exit (0);
}
```

rand_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 "rand.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int seed = 0;

void *
initialize_random_1_svc(long *argp, struct svc_req *rqstp)
{
  static char * result;
  seed += time(0) + 1; //we need to change the seed every time this is called
  srand((unsigned) seed);

  return (void *) &result;
}

double *
get_next_random_1_svc(void *argp, struct svc_req *rqstp)
{
  static double  result;

  result = (double)rand()/RAND_MAX*1.0;

  return &result;
}
```

Makefile.rand: (Has -ltirpc in LDLIBS)

```
# This is a template Makefile generated by rpcgen

# Parameters

CLIENT = rand_client
SERVER = rand_server

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

TARGETS_SVC.c = rand_svc.c rand_server.c
TARGETS_CLNT.c = rand_clnt.c rand_client.c
TARGETS = rand.h   rand_clnt.c rand_svc.c rand_client.c rand_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 -ltirpc
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)
```

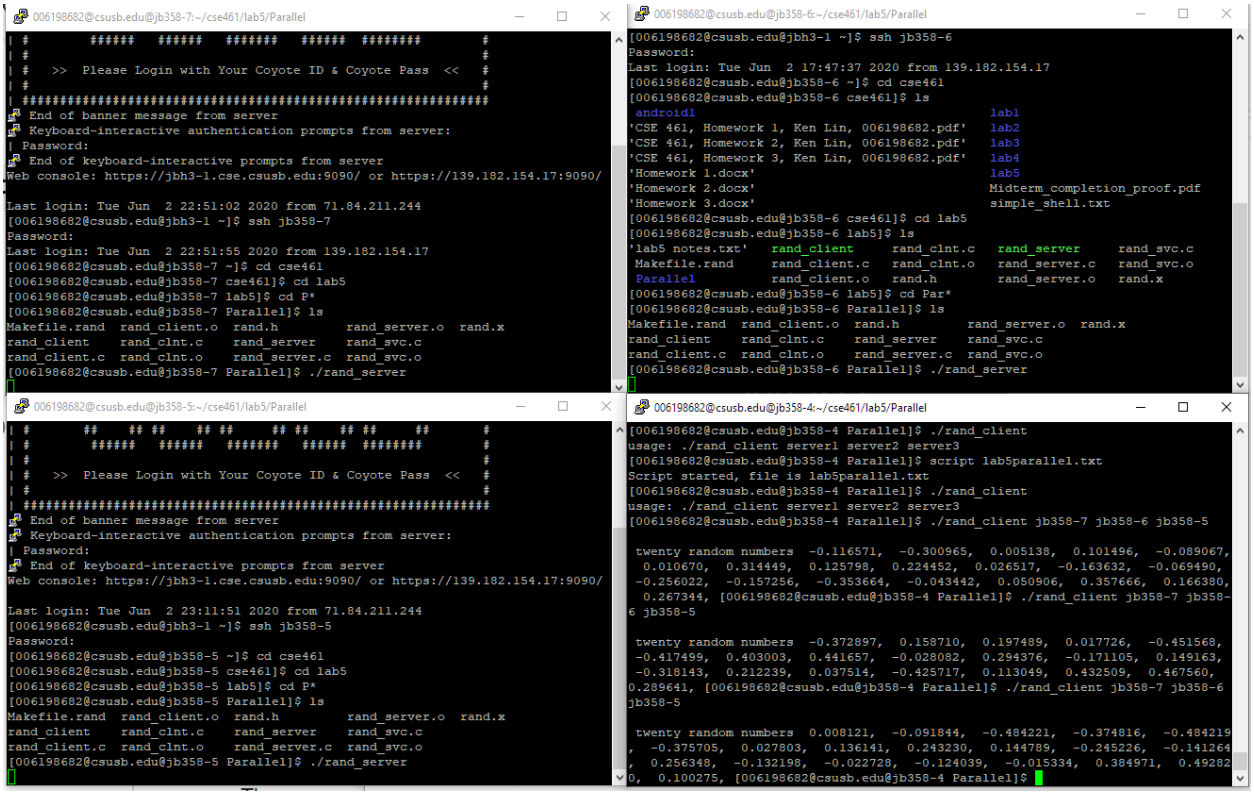
```
$(SERVER) : $(OBJECTS_SVC)
	$(LINK.c) -o $(SERVER) $(OBJECTS_SVC) $(LDLIBS)

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

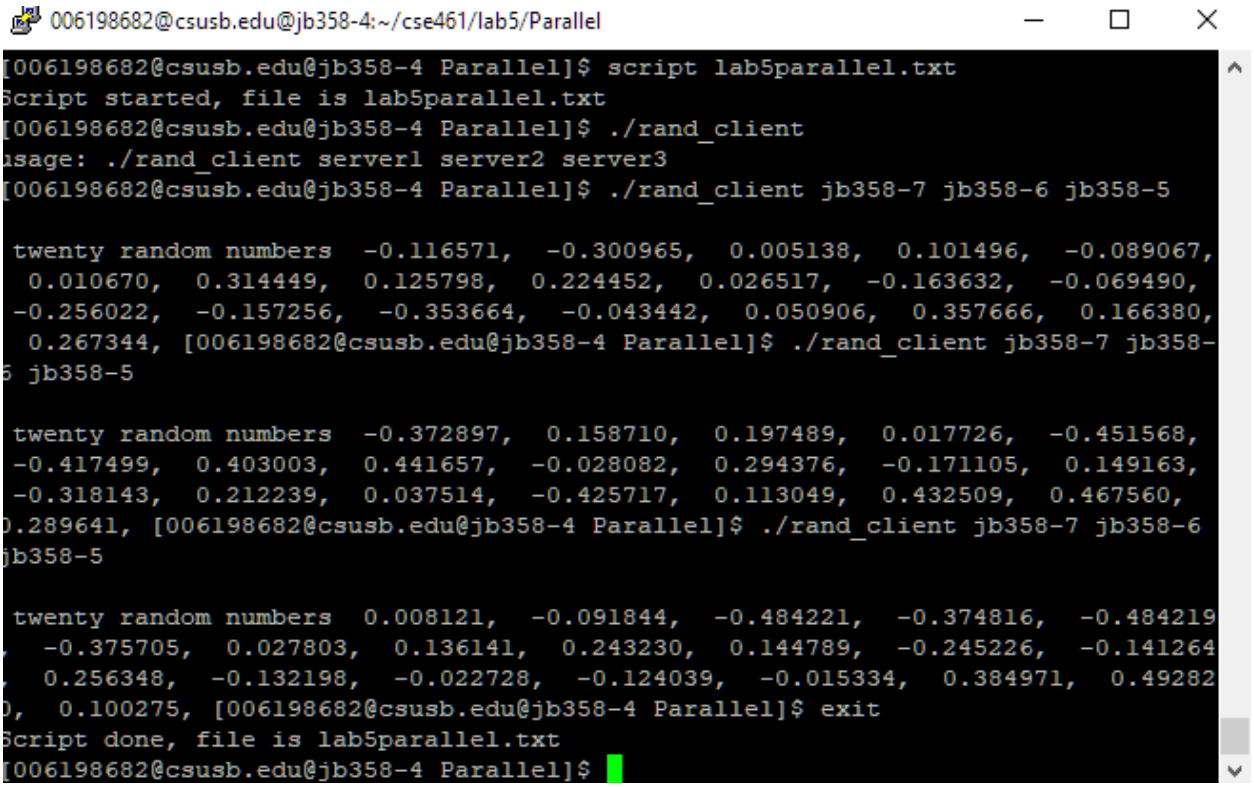
Part 9: (9 points assumed)

Parallel RNG

Screenshots:



Screenshot of all 3 servers + client running simultaneously, and also the client obtaining results.



Screenshot of client by itself.

Script file:

Of me running the client. (Servers were all initialized via ./rand_server)

lab5parallel.txt

```
Script started on 2020-06-02 23:22:51-07:00 [TERM="xterm" TTY="/dev/pts/0"
COLUMNS="80" LINES="24"]
   ]0;006198682@csusb.edu@jb358-
4:~/cse461/lab5/Parallel   [006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client
[006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client
usage: ./rand_client server1 server2 server3
   ]0;006198682@csusb.edu@jb358-
4:~/cse461/lab5/Parallel   [006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client jb358-7
jb358-6 jb358-5
[006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client jb358-7 jb358-6 jb358-5

 twenty random numbers  -0.116571,  -0.300965,  0.005138,  0.101496,  -0.089067,  0.010670,
0.314449,  0.125798,  0.224452,  0.026517,  -0.163632,  -0.069490,  -0.256022,  -0.157256,  -
0.353664,  -0.043442,  0.050906,  0.357666,  0.166380,  0.267344,
   ]0;006198682@csusb.edu@jb358-
```

```
4:~/cse461/lab5/Parallel   [006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client jb358-7
jb358-6 jb358-5
[006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client jb358-7 jb358-6 jb358-5

 twenty random numbers  -0.372897,  0.158710,  0.197489,  0.017726,  -0.451568,  -0.417499,
0.403003,  0.441657,  -0.028082,  0.294376,  -0.171105,  0.149163,  -0.318143,  0.212239,
0.037514,  -0.425717,  0.113049,  0.432509,  0.467560,  0.289641,
   ]0;006198682@csusb.edu@jb358-
4:~/cse461/lab5/Parallel   [006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client jb358-7
jb358-6 jb358-5
[006198682@csusb.edu@jb358-4 Parallel]$ ./rand_client jb358-7 jb358-6 jb358-5

 twenty random numbers  0.008121,  -0.091844,  -0.484221,  -0.374816,  -0.484219,  -
0.375705,  0.027803,  0.136141,  0.243230,  0.144789,  -0.245226,  -0.141264,  0.256348,  -
0.132198,  -0.022728,  -0.124039,  -0.015334,  0.384971,  0.492820,  0.100275,
   ]0;006198682@csusb.edu@jb358-
4:~/cse461/lab5/Parallel   [006198682@csusb.edu@jb358-4 Parallel]$       [Kexit
[006198682@csusb.edu@jb358-4 Parallel]$       [Kexit

Script done on 2020-06-02 23:24:26-07:00 [COMMAND_EXIT_CODE="0"]
```

Source Code:

Makefile.rand:

(Needed to edit this to enable math library functions.)

```
# This is a template Makefile generated by rpcgen

# Parameters

CLIENT = rand_client
SERVER = rand_server

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

TARGETS_SVC.c = rand_svc.c rand_server.c
TARGETS_CLNT.c = rand_clnt.c rand_client.c
```

```
TARGETS = rand.h   rand_clnt.c rand_svc.c rand_client.c rand_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 -ltirpc -lm
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)
```

Rand_clnt.c:

(Needed to edit this because client now sends two doubles to the server during the RPC.)

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#include <memory.h> /* for memset */
#include "rand.h"
```

```
/* Default timeout can be changed using clnt_control() */
static struct timeval TIMEOUT = { 25, 0 };

void *
initialize_random_1(long *argp, CLIENT *clnt)
{
        static char clnt_res;

        memset((char *)&clnt_res, 0, sizeof(clnt_res));
        if (clnt_call (clnt, INITIALIZE_RANDOM,
                (xdrproc_t) xdr_long, (caddr_t) argp,
                (xdrproc_t) xdr_void, (caddr_t) &clnt_res,
                TIMEOUT) != RPC_SUCCESS) {
                return (NULL);
        }
        return ((void *)&clnt_res);
}

double *
get_next_random_1(void *argp, CLIENT *clnt, double *x1, double *x2)
{
        static double clnt_res;

        memset((char *)&clnt_res, 0, sizeof(clnt_res));
        if (clnt_call (clnt, GET_NEXT_RANDOM,
                (xdrproc_t) xdr_void, (caddr_t) argp,
                (xdrproc_t) xdr_double, (caddr_t) &clnt_res,
                TIMEOUT) != RPC_SUCCESS) {
                return (NULL);
        }
        return (&clnt_res);
}
```

Rand.h:

(Same reason as above)

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#ifndef _RAND_H_RPCGEN
```

```
#define _RAND_H_RPCGEN

#include <rpc/rpc.h>


#ifdef __cplusplus
extern "C" {
#endif


#define RAND_PROG 6198682
#define RAND_VERS 1

#if defined(__STDC__) || defined(__cplusplus)
#define INITIALIZE_RANDOM 1
extern  void * initialize_random_1(long *, CLIENT *);
extern  void * initialize_random_1_svc(long *, struct svc_req *);
#define GET_NEXT_RANDOM 2
extern  double * get_next_random_1(void *, CLIENT *, double *, double *);
extern  double * get_next_random_1_svc(void *, struct svc_req *, double *, double *);
extern int rand_prog_1_freeresult (SVCXPRT *, xdrproc_t, caddr_t);

#else /* K&R C */
#define INITIALIZE_RANDOM 1
extern  void * initialize_random_1();
extern  void * initialize_random_1_svc();
#define GET_NEXT_RANDOM 2
extern  double * get_next_random_1();
extern  double * get_next_random_1_svc();
extern int rand_prog_1_freeresult ();
#endif /* K&R C */

#ifdef __cplusplus
}
#endif

#endif /* !_RAND_H_RPCGEN */
```

Rand_server.c:

(First of the ‘proper’ source code files)

```
/*
 * 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 "rand.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

int seed = 0;

void *
initialize_random_1_svc(long *argp, struct svc_req *rqstp)
{
  static char * result;
  seed += time(0) + 1; //we need to change the seed every time this is called
  srand((unsigned) seed);

  return (void *) &result;
}

double *
get_next_random_1_svc(void *argp, struct svc_req *rqstp, double *x1, double *x2)
{
  //to compute a random number with 3 servers, take as input two doubles
  //from up to two other machines
  //and return a random number generated here, plus the other two, modulo 1
  //That is, result = (random + x1 + x2) % 1
  //note that the distribution of number for this probably isn't uniform
  //that is, certain numbers should be more likely than others
  static double  result;

  result = (double)rand()/RAND_MAX*1.0;
  result = remainder((result + *x1 + *x2), 1.0); //can't use modulus directly

  return &result;
}
```

Rand_client.c:

(Lots of modifications to handle passing doubles to server & handling 3 servers.)

```
/*
```

```
 * 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 "rand.h"


double
rand_prog_1(char *host1, char *host2, char *host3)
{
        CLIENT *clnt1;
  CLIENT *clnt2;
  CLIENT *clnt3;

        void  *result_1;
        long  initialize_random_1_arg;
        double  *result_2;
        char *get_next_random_1_arg;
  double l, m;
  l = 0.0;
  m = 0.0;
  double *temp_1 = &l;
  double *temp_2 = &m;

#ifndef DEBUG
        clnt1 = clnt_create (host1, RAND_PROG, RAND_VERS, "udp");
        if (clnt1 == NULL) {
                clnt_pcreateerror (host1);
                exit (1);
        }

  clnt2 = clnt_create (host2, RAND_PROG, RAND_VERS, "udp");
        if (clnt2 == NULL) {
                clnt_pcreateerror (host2);
                exit (1);
        }

  clnt3 = clnt_create (host3, RAND_PROG, RAND_VERS, "udp");
        if (clnt3 == NULL) {
                clnt_pcreateerror (host3);
                exit (1);
        }
```

```
#endif  /* DEBUG */

        result_1 = initialize_random_1(&initialize_random_1_arg, clnt1);
        if (result_1 == (void *) NULL) {
                clnt_perror (clnt1, "call failed");
        }

        result_1 = initialize_random_1(&initialize_random_1_arg, clnt2);
        if (result_1 == (void *) NULL) {
                clnt_perror (clnt2, "call failed");
        }

  result_1 = initialize_random_1(&initialize_random_1_arg, clnt3);
        if (result_1 == (void *) NULL) {
                clnt_perror (clnt3, "call failed");
        }

  //not quite fully parallel because the client doesn't store earlier results
  //so instead what's happening is that server1 generates a random number by itself
        temp_1 = get_next_random_1((void*)&get_next_random_1_arg, clnt1, temp_1,
temp_2);
        if (temp_1 == (double *) NULL) {
                clnt_perror (clnt2, "call failed");
        }

  //server 2 generates a number from the results of server 1 and itself
  temp_2 = get_next_random_1((void*)&get_next_random_1_arg, clnt2, temp_1, temp_2);
        if (temp_2 == (double *) NULL) {
                clnt_perror (clnt3, "call failed");
        }

  //and server three generates a number from server 1, 2, and itself
  result_2 = get_next_random_1((void*)&get_next_random_1_arg, clnt3, temp_1, temp_2);
        if (result_2 == (double *) NULL) {
                clnt_perror (clnt3, "call failed");
        }


#ifndef DEBUG
        clnt_destroy (clnt1);
  clnt_destroy (clnt2);
  clnt_destroy (clnt3);
#endif   /* DEBUG */
```

```
  return *result_2;
}


int
main (int argc, char *argv[])
{
        char *host1;
  char *host2;
  char *host3;

        if (argc != 4) {
    //client needs to connect to three servers
                printf ("usage: %s server1 server2 server3\n", argv[0]);
                exit (1);
        }
        host1 = argv[1];
  host2 = argv[2];
  host3 = argv[3];

        rand_prog_1 (host1, host2, host3);

  double x;
     int i;
     printf("\n twenty random numbers ");
     for ( i = 0; i < 20; ++i ){
          x = rand_prog_1 (host1, host2, host3);
          printf(" %f, ", x );
     }
exit (0);
}
```