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.

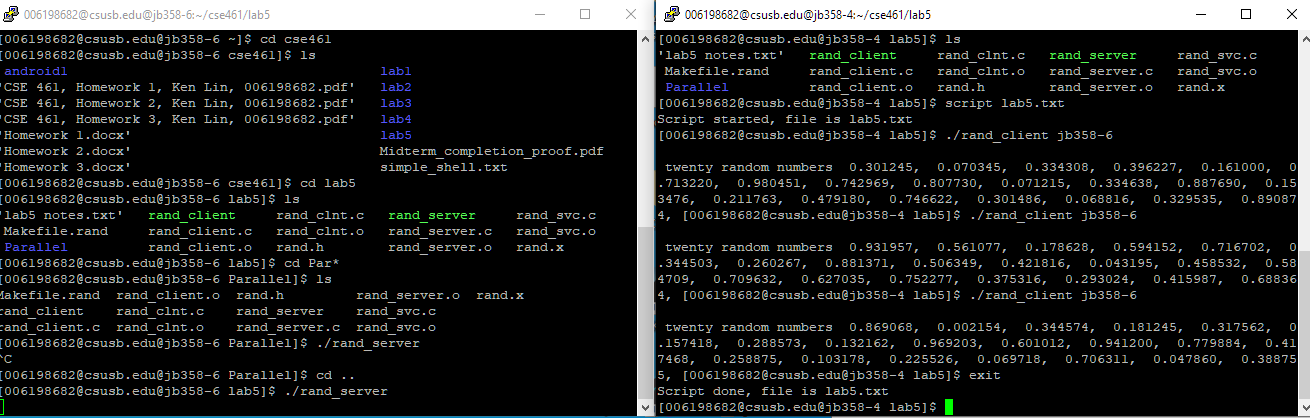
19 points.

**Report:**

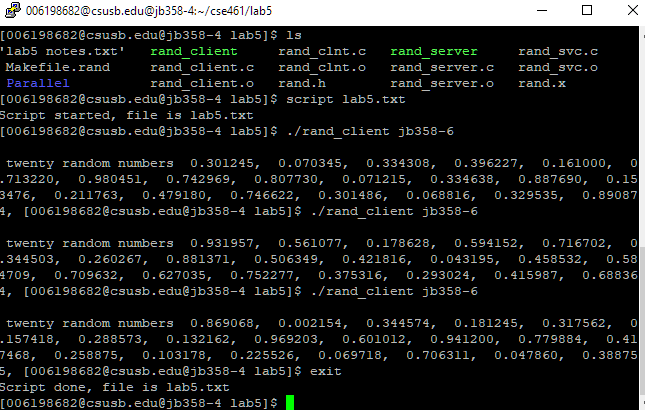
Parts 1 to 8: (10 points assumed)

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  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  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  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  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;  } |

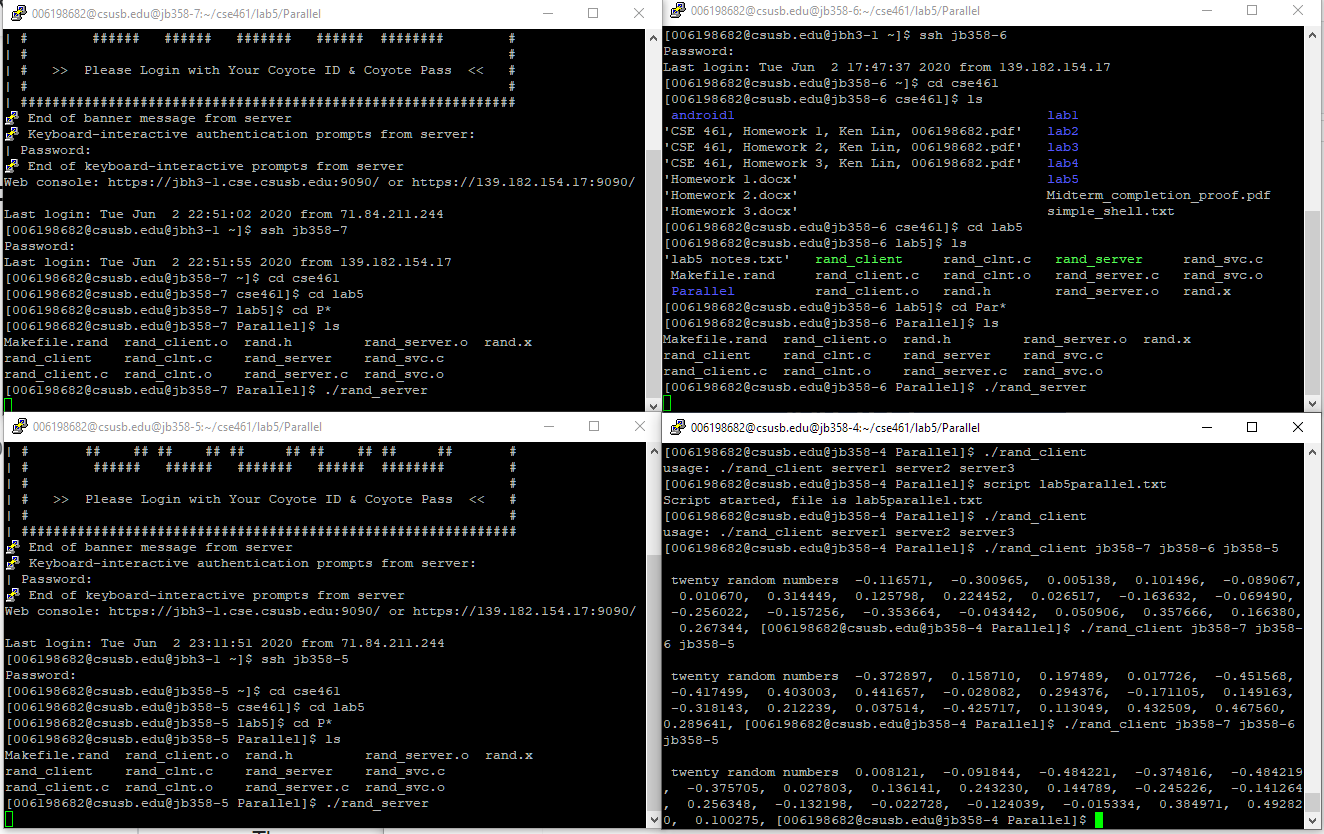
Makefile.rand: (Has -ltirpc in LDLIBS)

|  |
| --- |
| # 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) |

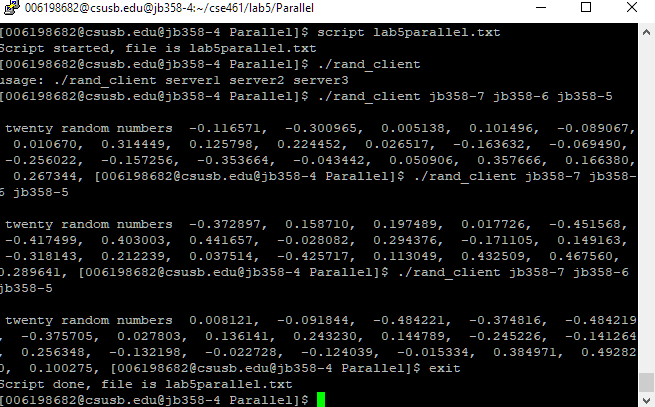
Part 9: (9 points assumed)

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  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  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  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  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  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);  } |