

CSCI 53700: Distributing Computing
Assignment Number 3

Date: November 16, 2018.

Submitted by: Anushka H Patil

Aim: To develop a simple distributed computing environment consisting of multiple Clients and a Server. The system is to be implemented in C or C++ and using the rpcgen utility discussed in the class. This assignment is intended to emphasize the RPC principles.

Remote Procedure Call (RPC)

According to the w3 standards, RPC is a technique for building distributed systems. Basically, it allows a program on one machine to call a subroutine on another machine without knowing that it is remote.

In simple terms it is a mechanism where one process(client) interacts with another process (server) on the remote host by means of procedure calls. Both processes are allocated stubs. This communication takes place as follows:

1. The client process makes a call to the client stub. It passes all the parameters referring to program local to it.
2. Marshalling takes place where all these parameters are marshalled and then a system call is made to transfer the marshalled packed.
3. The client-side RPC sends the packet over the network and is received by the RPC at the server end.
4. Unmarshalling takes place, where the original data i.e. parameters in our case are obtained from the marshalled content.
5. These parameters are then passed to the procedure for execution.
6. The result obtained is sent to the client in similar fashion.

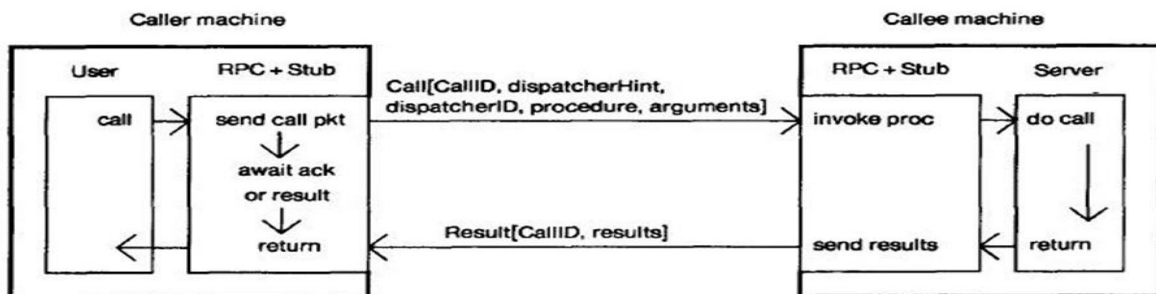


Image reference: <https://slideplayer.com/slide/5042760/>

rpcgen

It provides programmers a direct way to write distributed applications. The rpcgen protocol compiler accepts a remote program interface definition written in RPC language. It then produces C language output consisting of skeleton versions of the client routines, a server skeleton, XDR filter routines for both parameters and results, a header file that contains common definitions, and optionally, dispatch tables that the server uses to invoke routines that are based on authorization checks.

A3.x: It is the RPC Protocol Specification File which describes all the remote procedures listed below:

1. **hostname**: Returns the hostname on which the server is running.
2. **mergeSort**: Accepts two integer lists and returns their merged list that is sorted.
3. **encryptedEcho**: Returns an encrypted version (using any technique) of whatever a Client sends as an input.
4. **listFiles**: Returns a list of all files in the current directory.
5. **addComplex**: Accepts two complex numbers and returns their sum.

This file also contains different structures which are passed as parameters. Each of these above procedures have a procedure number. For instance, the *hostname* procedure is declared to be procedure 1, in version 1 of the remote program *A3PROG*, with program number *0x40000002*

The default output of rpcgen is:

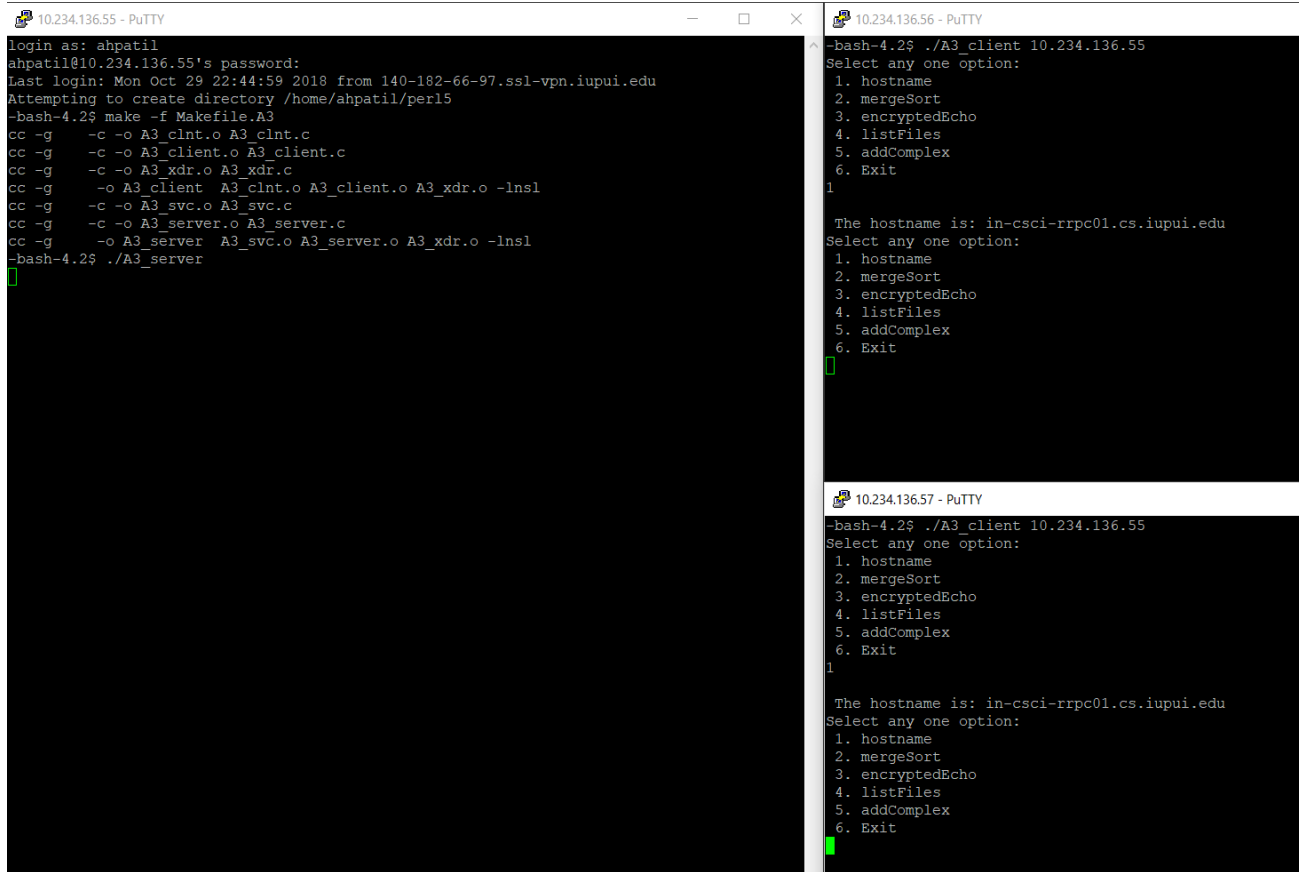
1. A header file of definitions common to the server and the client
2. A set of XDR routines that translate each data type defined in the header file
3. A stub program for the server
4. A stub program for the client

So, our A3.x file is then compiled using rpcgen to generate the header file (A3.h), client files (A3_clnt.c, A3_client.c), server files (A3_svc.c, A3_server.c), XDR file (A3_xdr.c), and makefile (Makefile.A3).

After the rpcgen protocol compilation, I modified the server-side and client-side code for each of the procedures as per the requirements then next step was to run the makefile using *make -f Makefile.A3* to compile the generated and modified files.

Using rpcgen, developing a distributed application is simple and easy. Developers can use it to generate sample client and server programs, and then modify them easily, according to requirements.

Concurrent access: multiple clients with concurrent access



```
10.234.136.55 - PuTTY
login as: ahpatil
ahpatil@10.234.136.55's password:
Last login: Mon Oct 29 22:44:59 2018 from 140-182-66-97.ssl-vpn.iupui.edu
Attempting to create directory /home/ahpatil/perl5
-bash-4.2$ make -f Makefile.A3
cc -g -c -o A3_clnt.o A3_clnt.c
cc -g -c -o A3_client.o A3_client.c
cc -g -c -o A3_xdr.o A3_xdr.c
cc -g -o A3_client A3_clnt.o A3_client.o A3_xdr.o -lnsl
cc -g -c -o A3_svc.o A3_svc.c
cc -g -c -o A3_server.o A3_server.c
cc -g -o A3_server A3_svc.o A3_server.o A3_xdr.o -lnsl
-bash-4.2$ ./A3_server

10.234.136.56 - PuTTY
-bash-4.2$ ./A3_client 10.234.136.55
Select any one Option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
1

The hostname is: in-csci-rrpc01.cs.iupui.edu
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit

10.234.136.57 - PuTTY
-bash-4.2$ ./A3_client 10.234.136.55
Select any one Option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
1

The hostname is: in-csci-rrpc01.cs.iupui.edu
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
```

Figure 1. server running on 10.234.136.55 (left) client 1 running on 10.234.136.56 (right top) and client 2 running on 10.234.136.57 (right bottom)

In figure 1, the server allows multiple clients to access concurrently. As we can see, both clients are connected to the server and have selected option 1 which returns the hostname on which the server is running. Thus, we can see that it returns *“The hostname is: in-csci-rrpc01.cs.iupui.edu”* which is the domain name for IP address 10.234.136.55 where the server is running. To explore more about the concurrent access, I tried to select different options by different clients.

The image displays three terminal windows from PuTTY, illustrating a concurrent server-client interaction. The leftmost window, titled '10.234.136.55 - PuTTY', shows the server's perspective: a user 'ahpatil' logs in, and the server compiles several C programs (A3_clnt.c, A3_client.c, A3_xdr.c, A3_svc.c, A3_server.c) using 'make -f Makefile.A3'. The middle window, titled '10.234.136.56 - PuTTY', shows client 1 running './A3_client 10.234.136.55'. It presents a menu with six options: 1. hostname, 2. mergeSort, 3. encryptedEcho, 4. listFiles, 5. addComplex, and 6. Exit. Option 2 is selected. The client then prompts for 'Input List Size for List 1' (value 1) and 'Input Integer 1: 22'. The rightmost window, titled '10.234.136.57 - PuTTY', shows client 2 running './A3_client 10.234.136.55'. It also presents the same menu, and option 3 is selected. The client prompts for 'Input String' (value 'abcd'). The server responds with 'The encryptedecho is: fghi'. Both client windows then show the menu again, indicating they can continue to interact with the server.

```
login as: ahpatil
ahpatil@10.234.136.55's password:
Last login: Mon Oct 29 22:44:59 2018 from 140-182-66-97.ssl-vpn.iupui.edu
Attempting to create directory /home/ahpatil/perl5
-bash-4.2$ make -f Makefile.A3
cc -g -c -o A3_clnt.o A3_clnt.c
cc -g -c -o A3_client.o A3_client.c
cc -g -c -o A3_xdr.o A3_xdr.c
cc -g -c -o A3_client A3_clnt.o A3_client.o A3_xdr.o -lnsl
cc -g -c -o A3_svc.o A3_svc.c
cc -g -c -o A3_server.o A3_server.c
cc -g -c -o A3_server A3_svc.o A3_server.o A3_xdr.o -lnsl
-bash-4.2$ ./A3_server

-bash-4.2$ ./A3_client 10.234.136.55
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
2

Input List Size for List 1
1

Input Integer 1: 22

Input List Size for List 2

-bash-4.2$ ./A3_client 10.234.136.55
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
3

Input String
abcd

The encryptedecho is: fghi
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
```

Figure 2. server running on 10.234.136.55 (left) client 1 running on 10.234.136.56 (right top) and client 2 running on 10.234.136.57 (right bottom)

In figure 2, the client 1 (running on 10.234.136.56) and client 2 (running on 10.234.136.57) select option 2 and option 3 respectively. The option 2 is for mergesort while option 3 is for encryptedecho. Client 2 receives the response “*The encryptedecho is: fghi*” and concurrently Client 1 is entering the required inputs for merge sorting. This clearly indicates that the design system provides concurrent access to multiple clients.

The server can serve multiple clients simultaneously without any miscommunication or mismanagement.

Run Samples

1. rpcgen compilation

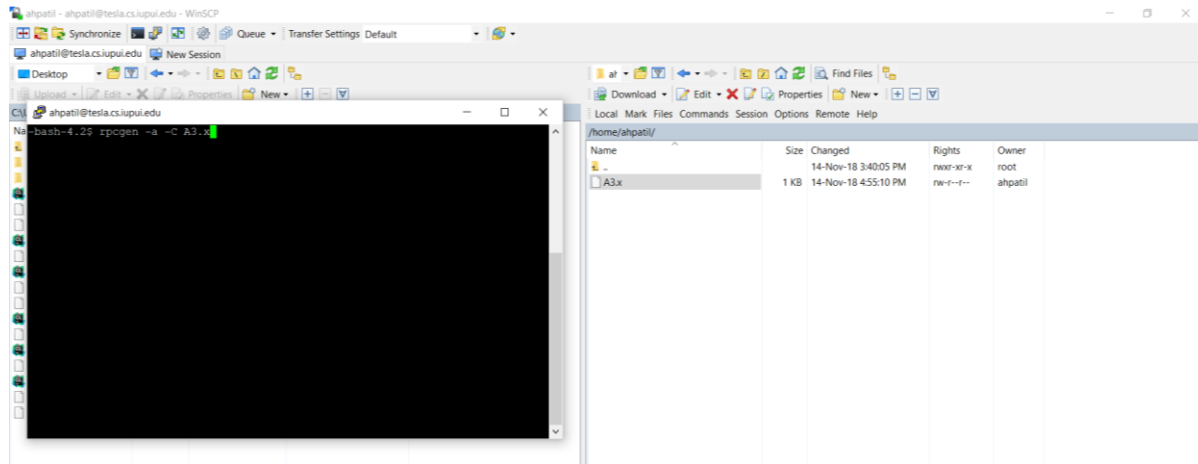


Figure 3. Before the rpcgen is compiled

Pre-compilation, there is only a single file present in the directory which is the A3.x file. The compilation command is entered *rpcgen -a -C A3.x*

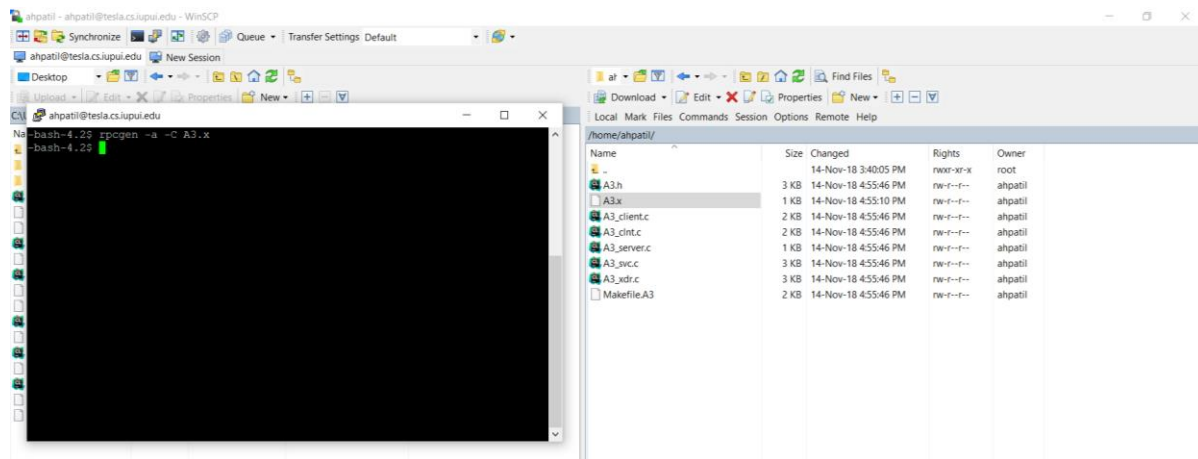
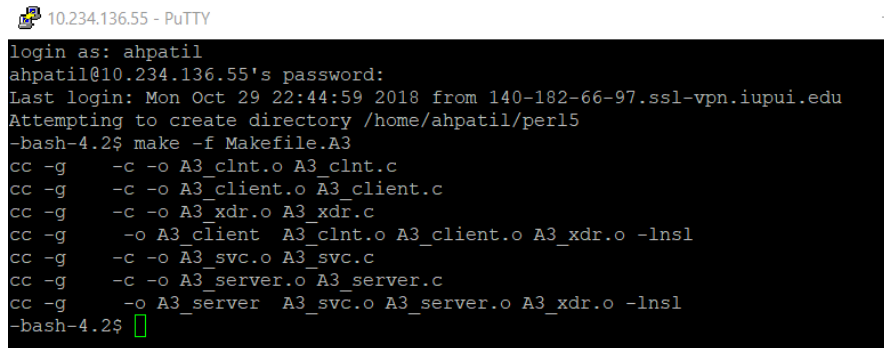


Figure 4. After the rpcgen is compiled

Post-compilation, it generated the following files: the header file (A3.h), client files (A3_clnt.c, A3_client.c), server files (A3_svc.c, A3_server.c), XDR file (A3_xdr.c), and makefile (Makefile.A3).

2. Makefile

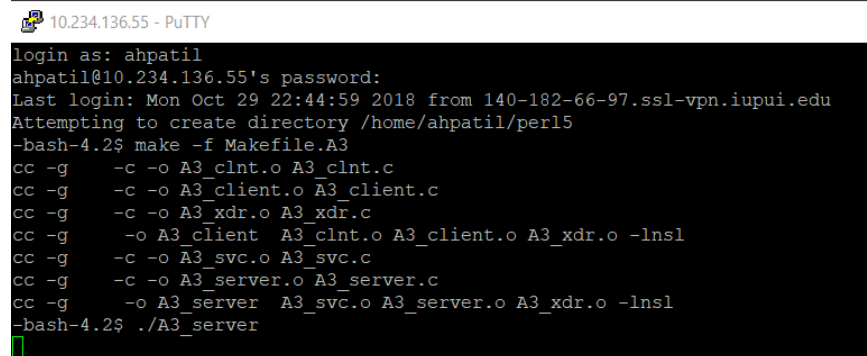


```
10.234.136.55 - PuTTY
login as: ahpatil
ahpatil@10.234.136.55's password:
Last login: Mon Oct 29 22:44:59 2018 from 140-182-66-97.ssl-vpn.iupui.edu
Attempting to create directory /home/ahpatil/perl5
-bash-4.2$ make -f Makefile.A3
cc -g      -c -o A3_clnt.o A3_clnt.c
cc -g      -c -o A3_client.o A3_client.c
cc -g      -c -o A3_xdr.o A3_xdr.c
cc -g      -o A3_client  A3_clnt.o A3_client.o A3_xdr.o -lnsl
cc -g      -c -o A3_svc.o A3_svc.c
cc -g      -c -o A3_server.o A3_server.c
cc -g      -o A3_server  A3_svc.o A3_server.o A3_xdr.o -lnsl
-bash-4.2$
```

Figure 5. Running the makefile

The following command is used to execute the makefile *make -f Makefile.A3*

3. Running the server on 10.234.136.55



```
10.234.136.55 - PuTTY
login as: ahpatil
ahpatil@10.234.136.55's password:
Last login: Mon Oct 29 22:44:59 2018 from 140-182-66-97.ssl-vpn.iupui.edu
Attempting to create directory /home/ahpatil/perl5
-bash-4.2$ make -f Makefile.A3
cc -g      -c -o A3_clnt.o A3_clnt.c
cc -g      -c -o A3_client.o A3_client.c
cc -g      -c -o A3_xdr.o A3_xdr.c
cc -g      -o A3_client  A3_clnt.o A3_client.o A3_xdr.o -lnsl
cc -g      -c -o A3_svc.o A3_svc.c
cc -g      -c -o A3_server.o A3_server.c
cc -g      -o A3_server  A3_svc.o A3_server.o A3_xdr.o -lnsl
-bash-4.2$ ./A3_server
```

Figure 6. Running the server on 10.234.136.55

The following command is used to run the server. In my case the server is running on 10.234.136.55 the domain name for which is in-csci-rrpc01.cs.iupui.edu.

4. Client running on 10.234.136.56

The client running on 10.234.136.56 performs of the following tasks

- a. **hostname** – Returns the hostname on which the server is running



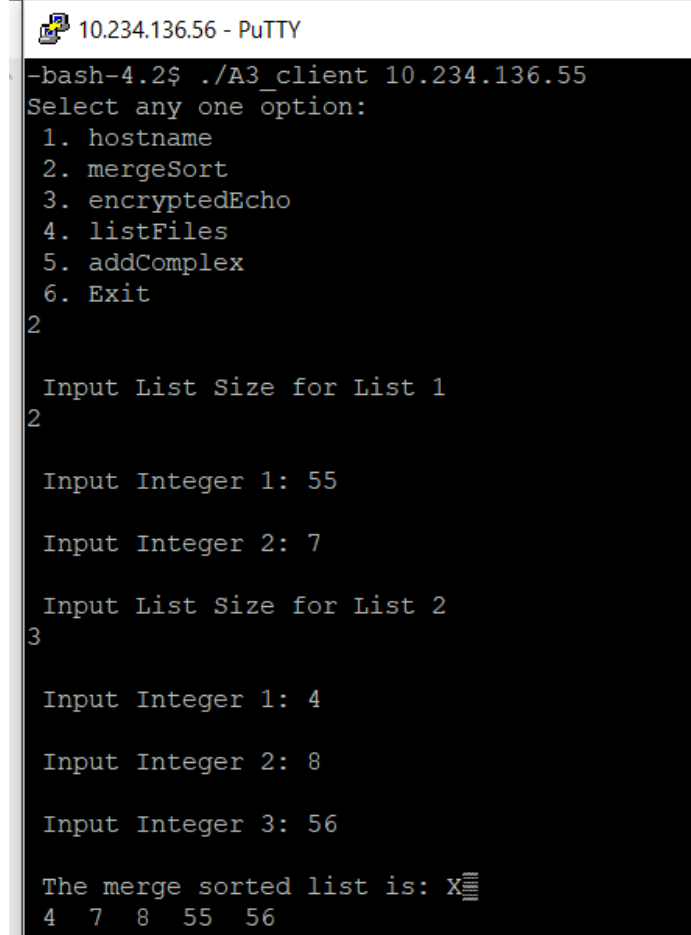
```
10.234.136.56 - PuTTY
-bash-4.2$ ./A3_client 10.234.136.55
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
1

The hostname is: in-csci-rrpc01.cs.iupui.edu
```

Figure 7. clients select option 1

The server sends the following response “The hostname is: in-csci-rrpc01.cs.iupui.edu” which is the domain name of the machine on which the server is running.

b. mergeSort – Accepts two integer lists and returns their merged list that is sorted.



```
10.234.136.56 - PuTTY
-bash-4.2$ ./A3_client 10.234.136.55
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
2

Input List Size for List 1
2

Input Integer 1: 55

Input Integer 2: 7

Input List Size for List 2
3

Input Integer 1: 4

Input Integer 2: 8

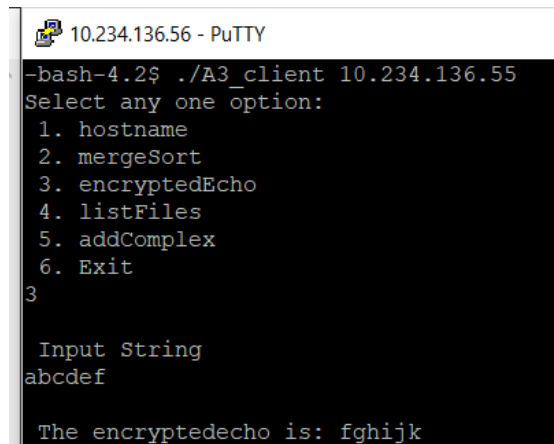
Input Integer 3: 56

The merge sorted list is: X
4 7 8 55 56
```

Figure 8. clients select option 2

The client is asked to input the two integer lists. Client enters list 1: 55, 7 and list 2: 4, 8, 56. The server on reception merges these two lists and sorts it in ascending order and returns a response “*The merge sorted list is: 4 7 8 55 56*”

c. encryptedEcho – Returns an encrypted version of whatever a Client sends as an input.



```
10.234.136.56 - PuTTY
-bash-4.2$ ./A3_client 10.234.136.55
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
3

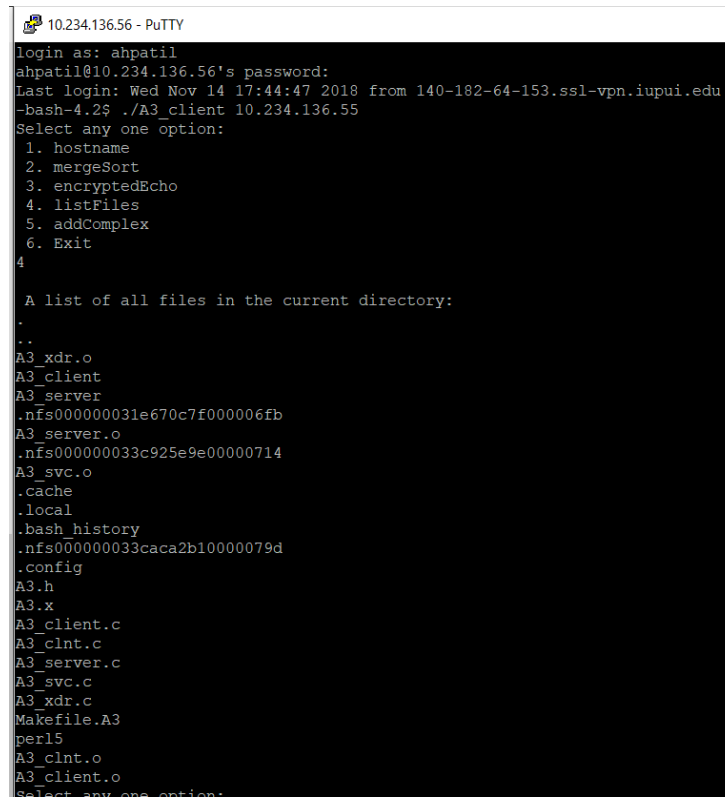
Input String
abcdef

The encryptedecho is: fghijk
```

Figure 9. clients select option 3

The client is asked to input the string. Client enters string *abcdef*. The server on reception of the string encrypts it and responds with the following response “*The encryptedecho is: fghijk*” The encryption mechanism that I have used here uses key value 5, which is added to the ASCII value of the characters. This mechanism works fine as we can see that the first character *a* is shifted by 5 characters thus the encrypted strings first character is *f*.

d. listFiles – Returns a list of all files in the current directory.



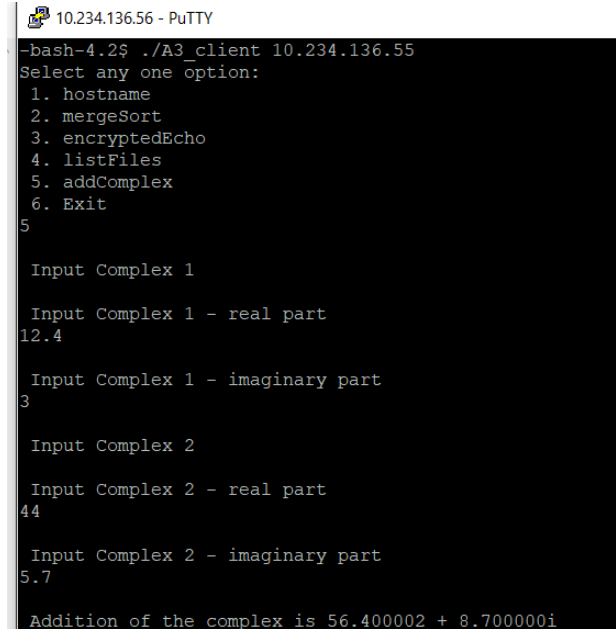
```
10.234.136.56 - PuTTY
login as: ahpatil
ahpatil@10.234.136.56's password:
Last login: Wed Nov 14 17:44:47 2018 from 140-182-64-153.ssl-vpn.iupui.edu
-bash-4.2$ ./A3_client 10.234.136.55
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
4

A list of all files in the current directory:
.
..
A3_xdr.o
A3_client
A3_server
.nfs0000000031e670c7f000006fb
A3_server.o
.nfs0000000033c925e9e00000714
A3_svc.o
.cache
.local
.bash_history
.nfs0000000033caca2b10000079d
.config
A3.h
A3.x
A3_client.c
A3_clnt.c
A3_server.c
A3_svc.c
A3_xdr.c
Makefile.A3
perl5
A3_clnt.o
A3_client.o
Select any one option:
```

Figure 10. clients select option 4

The client selects option 4 which list's the files in the current directory. The server responds with a list of files in the directory which can be clearly noticed in the figure above.

e. addComplex – Accepts two complex numbers and returns their sum



```
10.234.136.56 - PuTTY
-bash-4.2$ ./A3_client 10.234.136.55
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
5

Input Complex 1

Input Complex 1 - real part
12.4

Input Complex 1 - imaginary part
3

Input Complex 2

Input Complex 2 - real part
44

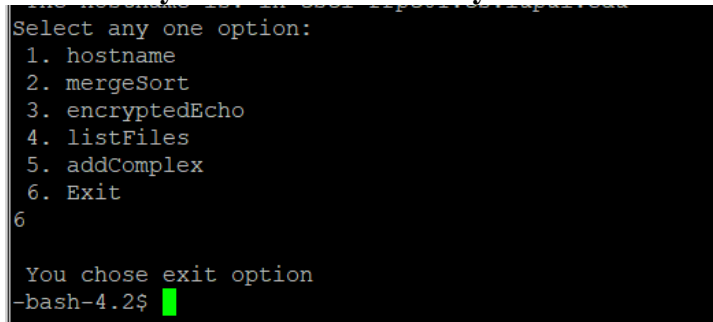
Input Complex 2 - imaginary part
5.7

Addition of the complex is 56.400002 + 8.700000i
```

Figure 11. clients select option 5

The client selects option 5 which adds the complex number. The client inputs the two complex numbers: $12.4 + 3i$ and $44 + 5.7i$. The sever on reception of these values, executes the addComplex function and returns the following response “Addition of the complex is $56.400002 + 8.700000i$ ”

f. Exit – This option is used by the client to exit the system



```
Select any one option:
1. hostname
2. mergeSort
3. encryptedEcho
4. listFiles
5. addComplex
6. Exit
6

You chose exit option
-bash-4.2$
```

Figure 12. clients select option 6

The client selects option 6 to exit the system.

Design Decisions:

1. At the client side, I have used a while loop which is true. This while loop consist of the task options that a client can select. The while loop breaks either when the client selects option 6. Exit or if any client-side error occurs. This way if client plans to execute 2 task one after another s/he can continue the task execution.
2. Initially when I was designing the model, I had few issues with the character array which resulted in the segmentation faults, so I decided to define a string type which helped me to get rid of this segmentation fault error. So, I have defined the *typedef string s<10000>* in the A3.x file.
3. Encryption mechanism is very simple yet effective. The key value '5' is added to the ASCII of the characters.

Pros of using RPC:

1. It is server independent
2. RPC supports both process-oriented and thread-oriented models
3. rpcgen makes it easy and simple to develop the distributed systems.
4. The procedure calls preserve the business logics which is apt for the application.

Cons of using RPC:

1. RPC is not a standard but an idea that is implemented in many ways.
2. Scheduling cost is increased due to context switching.

Conclusion:

Thus, I have successfully developed a simple distributed computing environment consisting of multiple Clients and a Server. The system is implemented in C, using the rpcgen utility and the RPC principles. The system provides concurrent access to multiple clients thus, successfully serving multiple clients simultaneously.

References:

1. Lecture slides
2. Wikipedia
3. w3.org
4. Tutorial points
https://www.tutorialspoint.com/data_communication_computer_network/client_server_model.htm
5. Oracle documentation <https://docs.oracle.com/cd/E19683-01/816-1435/rpcgenpguide-21470/index.html>