Indiana University-Purdue University Indianapolis

Department of Computer and Information Science

CSCI 53700

Fall 2017

Assignment - 2

AKHIL NAYABU
ID: 2000075395

Table of Contents

I. Introduction

The main objective of this assignment is to build a distributed system using remote procedure call concepts in C. In this assignment, we build a server and multiple clients which interact through remote procedure calls.

II. RPC Concepts & Design

RPC stands for remote procedure call. RPC is typically used to build distributed client-server applications. In a normal local procedure call the caller and the procedure called should reside in the same address space, but whereas in RPC these two need exists in the same address space.

Each remote procedure is uniquely identified by a combination of program number, version number and procedure number. We can have multiple version of a remote procedure made available to the clients. All the remote procedures are grouped by the program number. When a client invokes a remote procedure call, the client thread is put on hold and the request is sent to the server. The server once it receives the request, the server calls the routine which was requested. After execution of this procedure on the server, server sends back the response to client. The client program receives the response and continues with its execution. (Marshall, 1999)

For us to develop an RPC client server distributed application; we need to first define the protocol. Protocol contains the name of the procedure, data types of parameters and return types. Protocol is defined in an .x file (In our assignment it is add.x). Once we have defined the protocol, we execute rpcgen command with –A and –C flags. This would generate the stubs for server (add_server.c) and client (add_client.c). We have used two compile time flags –A and –C where –A is called as "MT auto mode" which by default enables multi-threading and –C is used to generate the stubs in ANSI C. Once stubs are generated we need to add the processing logic in both server and client. (Compile-Time Flags, 2017)

In our assignment, we have three main files:

- **Add.x:** this file contains the protocol. It contains the definition of the four-remote procedure that were to be implemented as part of this assignment.
- **Add_server.c:** This file has the code for server. We have the implementation for the four remote procedures for this assignment. These four functions include:
    - getpath_1_svc: This procedure will return the current working directory of the server. This is fulfilled by using the getcwd() method from the standard C library.
    - isfileexist_1_svc: This procedure identifies if the requested file is present in the directory or not.
    - printecho_1_svc: This procedure will echo the message sent by client back to it.

  - mmultiply_1_svc: This procedure performs matrix multiplication
  - sort_1_svc: This procedure will sort the array passed and returns the sorted array.

- Add_client.c: This C file contains the menu to invoke various remote procedures from server.

III. Pros & Cons of RPC

A. Pros

i. Easy to build a distributed system since it uses semantics which are straight forward.

ii. Preserves business logic in application development

iii. Server independent

iv. Supports process-oriented and thread-oriented models.

B. Cons

i. Poor error handling

ii. Context switching increases scheduling costs

iii. No standard way to implement RPC, can be implemented in many ways which differ

iv. It is not hardware independent.

v. Only works over networks which use TCP/IP protocol.

IV. Functionality and Discussion

A. **Get Path**: gets current working directory on which the server is running. Used getcwd() function from unistd.h library for which character buffer and size of buffer is sent as arguments.

B. **Echo**: takes a message from client passes to server and returns it back to client. Prints the returned string on client.

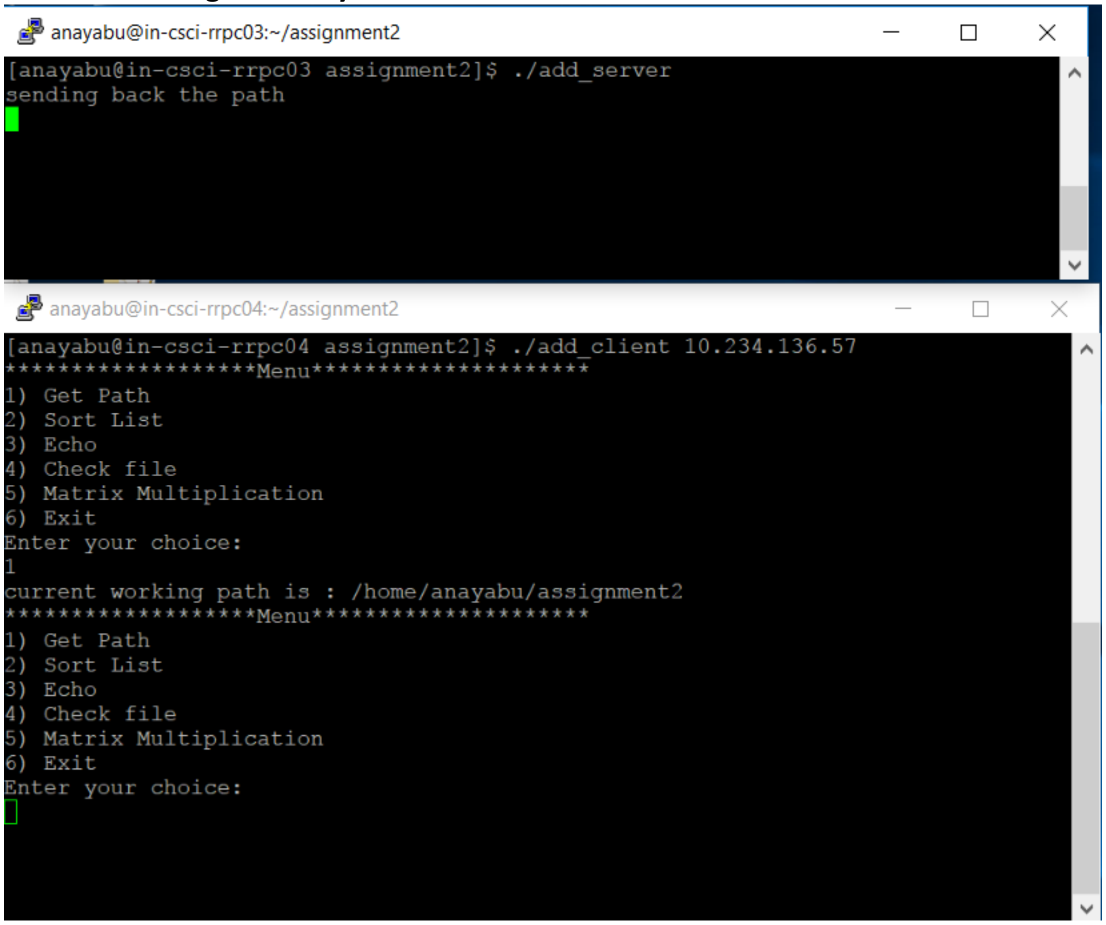
C. **File Status**: takes input file name from user and checks with a file with given name exists in the current directory. Returns “File found” if it exists, “File not found” if it does not exists

D. **Sort**: takes size and elements of list as input, passes it to server where server sorts the given list and returns resultant list which is printed on client.

E. **Matrix Multiplication**: takes in row and column as input, to make is simple I have considered both are matrices of same dimensions. Passes matrices and dimensions to server and server computes the multiplication of these two matrices and returns the resultant matrix back which is printed on Client.

V. Sample Runs (Screenshots)

Current working Directory:



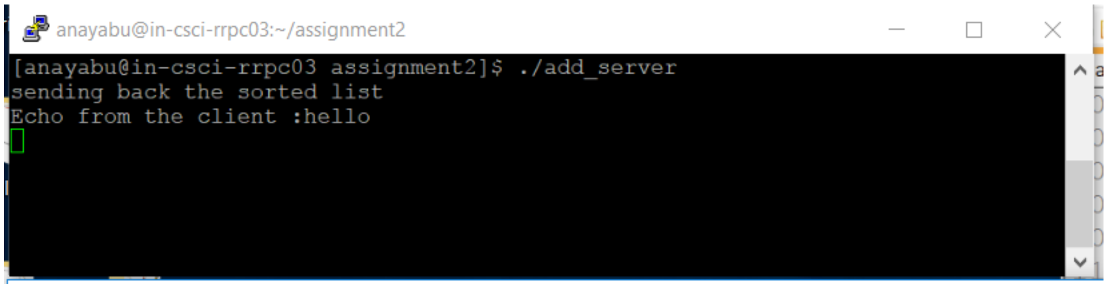
anayabu@in-csci-rrpc03:~/assignment2

```
[anayabu@in-csci-rrpc03 assignment2]$ ./add_server
sending back the path
```

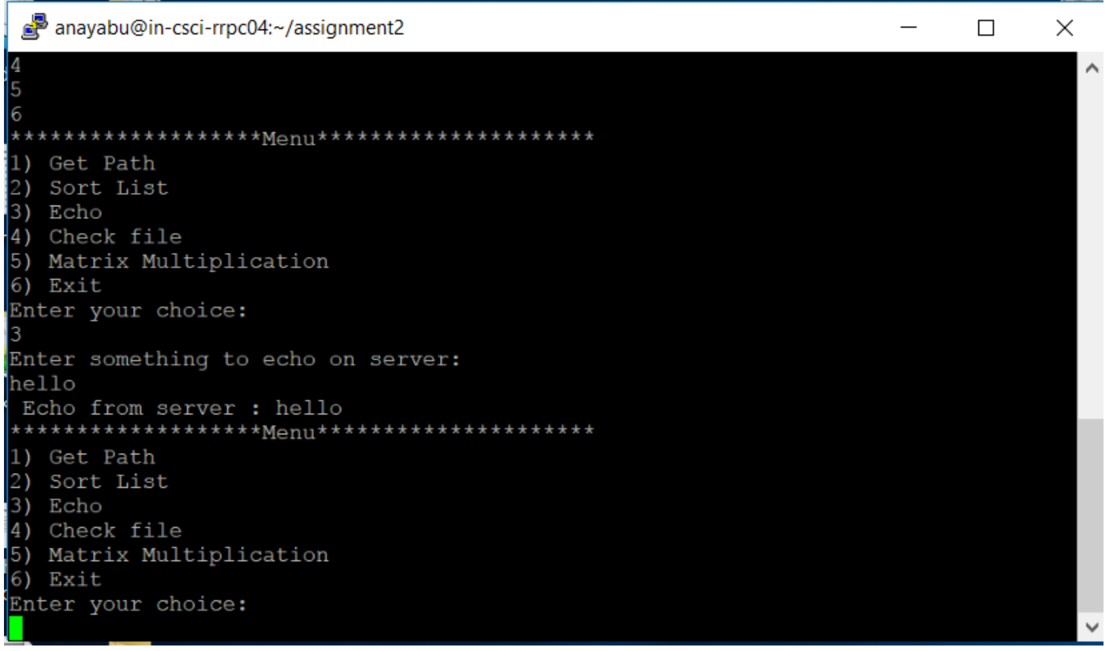
anayabu@in-csci-rrpc04:~/assignment2

```
[anayabu@in-csci-rrpc04 assignment2]$ ./add_client 10.234.136.57
******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
1
current working path is : /home/anayabu/assignment2
******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
```

Echo:

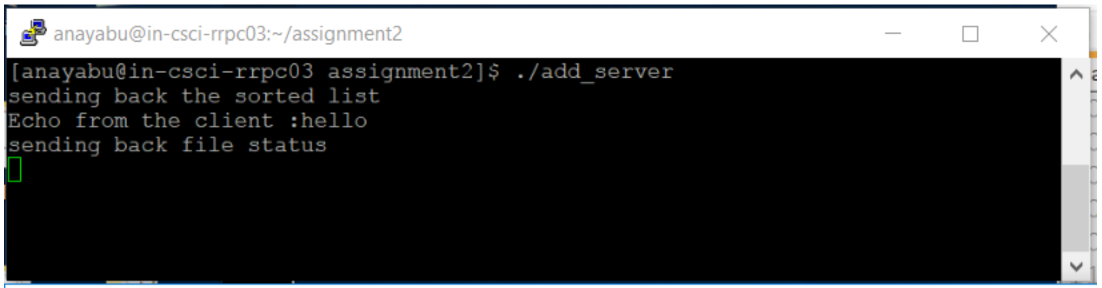


```
[anayabu@in-csci-rrpc03 assignment2]$ ./add_server
sending back the sorted list
Echo from the client :hello
```

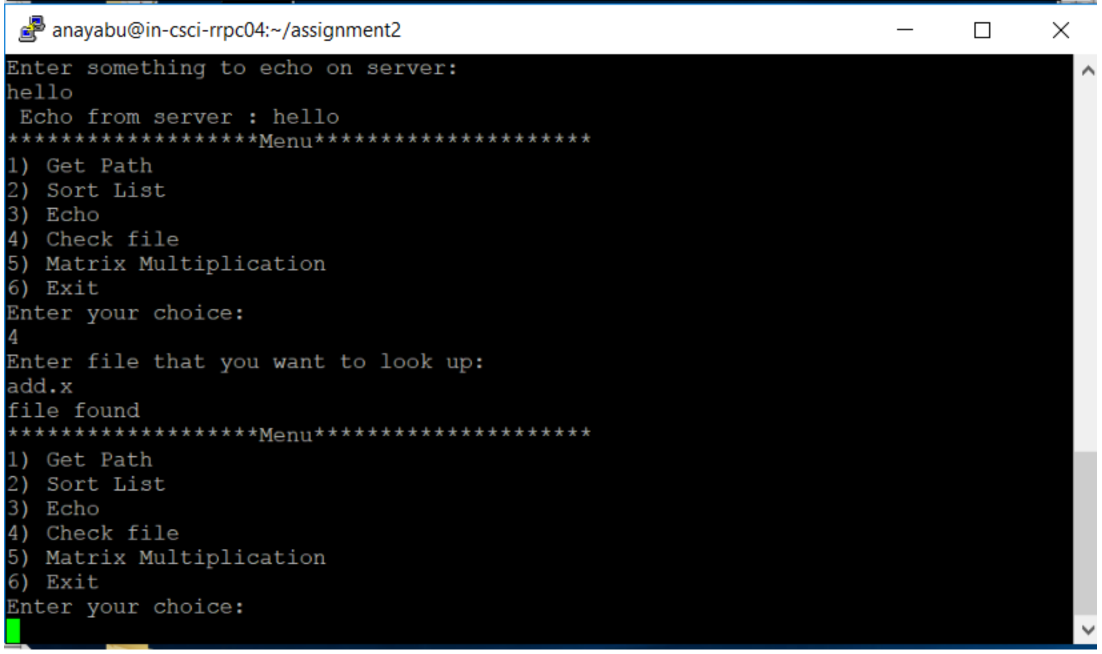


```
4
5
6
******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
3
Enter something to echo on server:
hello
 Echo from server : hello
******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
```

File Status:

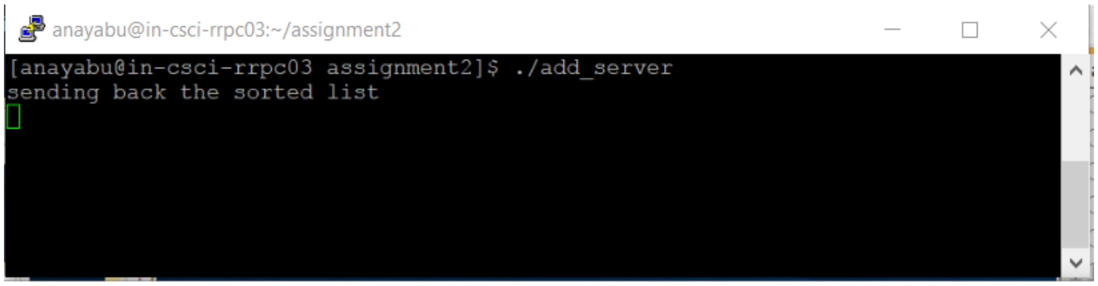

anayabu@in-csci-rrpc03:~/assignment2

```
[anayabu@in-csci-rrpc03 assignment2]$ ./add_server
sending back the sorted list
Echo from the client :hello
sending back file status
```


anayabu@in-csci-rrpc04:~/assignment2

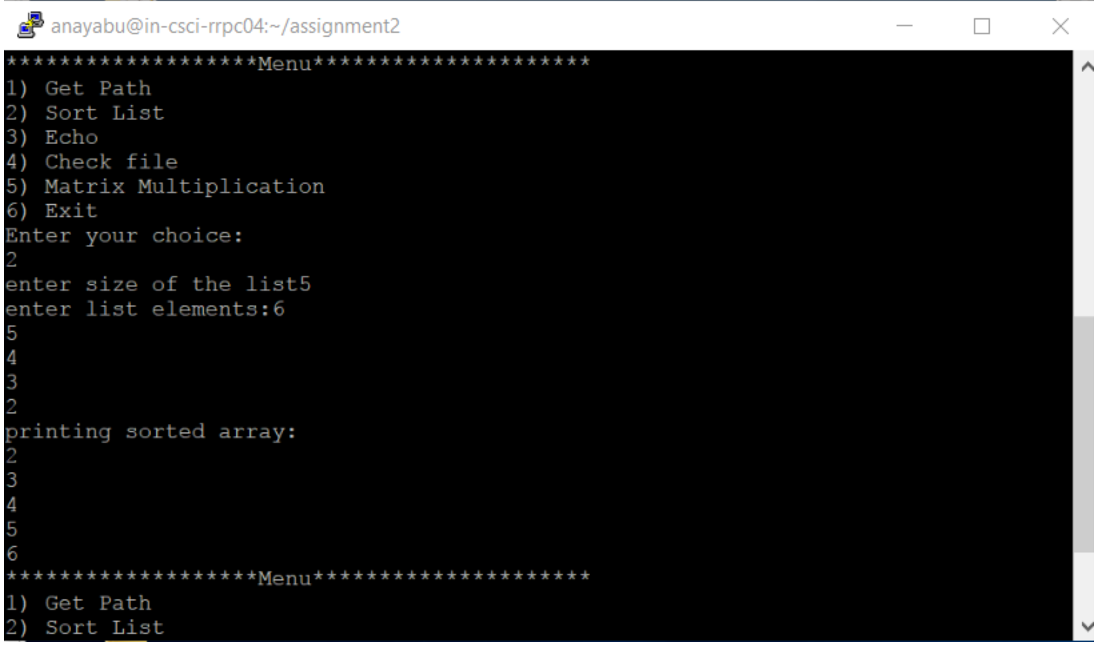
```
Enter something to echo on server:
hello
 Echo from server : hello
*******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
4
Enter file that you want to look up:
add.x
file found
*******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
```

Sort List:



anayabu@in-csci-rrpc03:~/assignment2

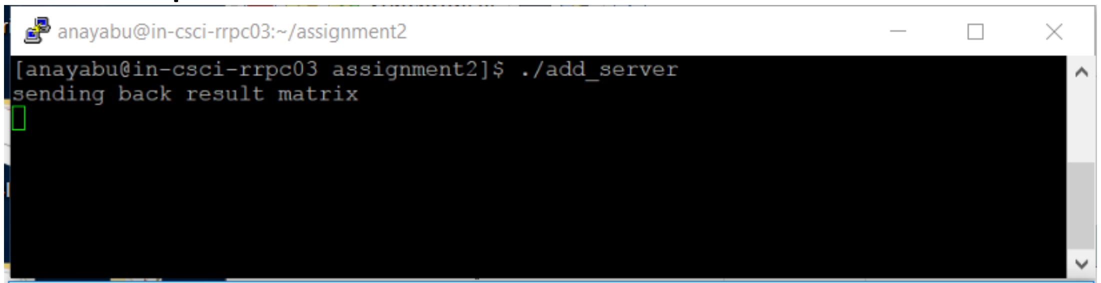
```
[anayabu@in-csci-rrpc03 assignment2]$ ./add_server
sending back the sorted list
```



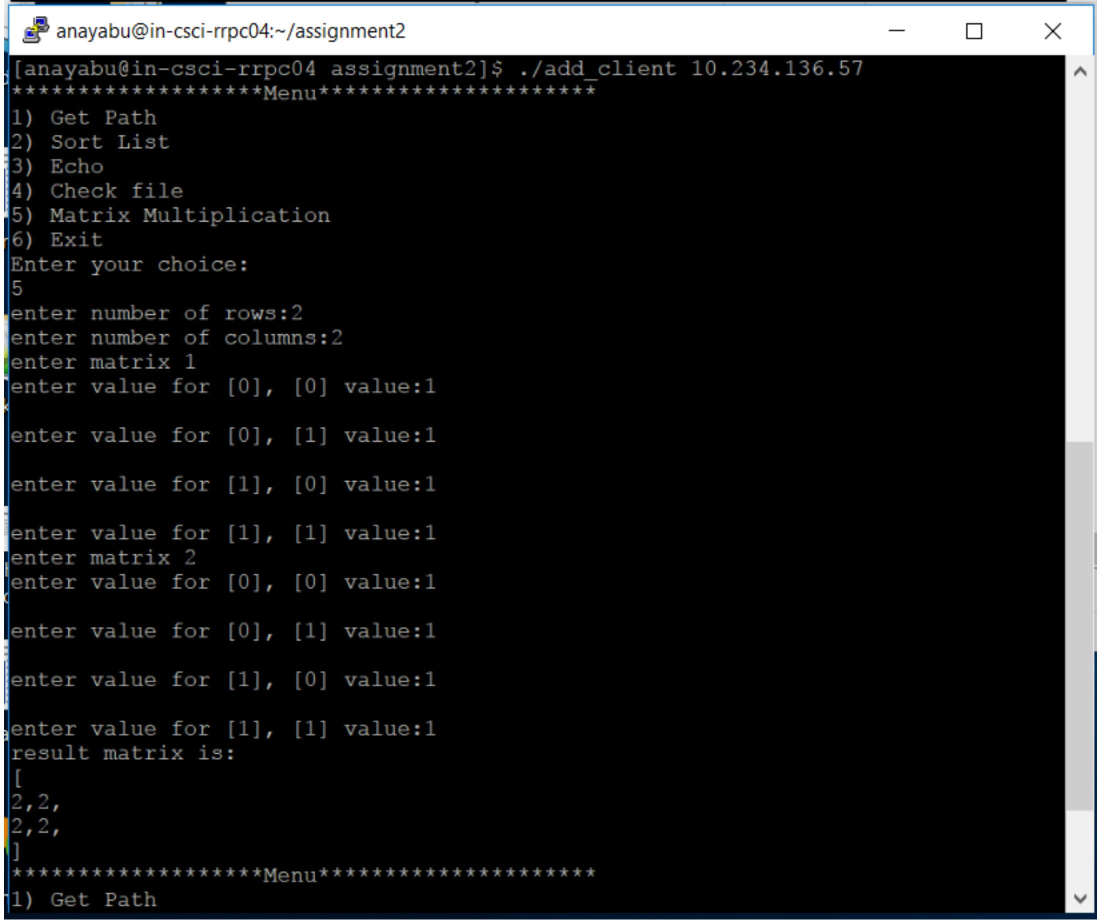
anayabu@in-csci-rrpc04:~/assignment2

```
******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
2
enter size of the list5
enter list elements:6
5
4
3
2
printing sorted array:
2
3
4
5
6
******************Menu*********************
1) Get Path
2) Sort List
```

Matrix Multiplication:

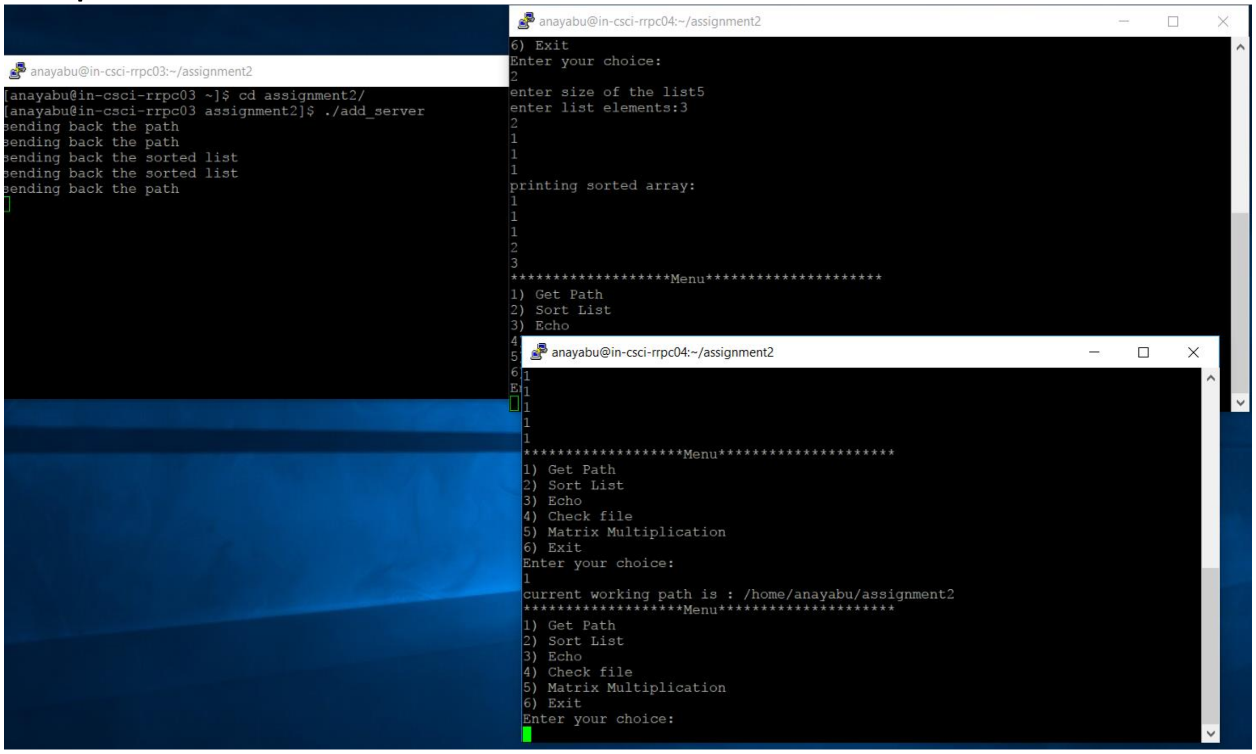


```
[anayabu@in-csci-rrpc03 assignment2]$ ./add_server
sending back result matrix
```



```
[anayabu@in-csci-rrpc04 assignment2]$ ./add_client 10.234.136.57
*******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
5
enter number of rows:2
enter number of columns:2
enter matrix 1
enter value for [0], [0] value:1

enter value for [0], [1] value:1

enter value for [1], [0] value:1

enter value for [1], [1] value:1
enter matrix 2
enter value for [0], [0] value:1

enter value for [0], [1] value:1

enter value for [1], [0] value:1

enter value for [1], [1] value:1
result matrix is:
[
2,2,
2,2,
]
*******************Menu*********************
1) Get Path
```

Multiple Client Connections:



anayabu@in-csci-rrpc03:~/assignment2

```
[anayabu@in-csci-rrpc03 ~]$ cd assignment2/
[anayabu@in-csci-rrpc03 assignment2]$ ./add_server
sending back the path
sending back the path
sending back the sorted list
sending back the sorted list
sending back the path
```

anayabu@in-csci-rrpc04:~/assignment2

```
6) Exit
Enter your choice:
2
enter size of the list5
enter list elements:3
2
1
1
1

printing sorted array:
1
1
1
2
3
*******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4
5
6
E
```

anayabu@in-csci-rrpc04:~/assignment2

```
1
1
1
1
1
*******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
1
current working path is : /home/anayabu/assignment2
*******************Menu*********************
1) Get Path
2) Sort List
3) Echo
4) Check file
5) Matrix Multiplication
6) Exit
Enter your choice:
```