Assignment - 6

Operating Systems

Question 2

Two processes can communicate using shared memory. Similarly they can communicate using a 'socket'.

Socket based inter-process communication is more powerful than shared memory based communication as it gives you the power to enable communication between two processes running even on two different physical computers (connected by a LAN/WLAN). Study the pages p. No. 48 - p. No. 51 of Unix System Programming Part-II ("Prog-II.pdf") document to learn socket programming.

Now, implement a client-server project where the client and server should communicate using sockets. Server should provide a computing service which a client / clients can request.

Service: Calculate the LCM of 4 given integers

Group - 6

Abhijeet Sonkar
IIB2019009

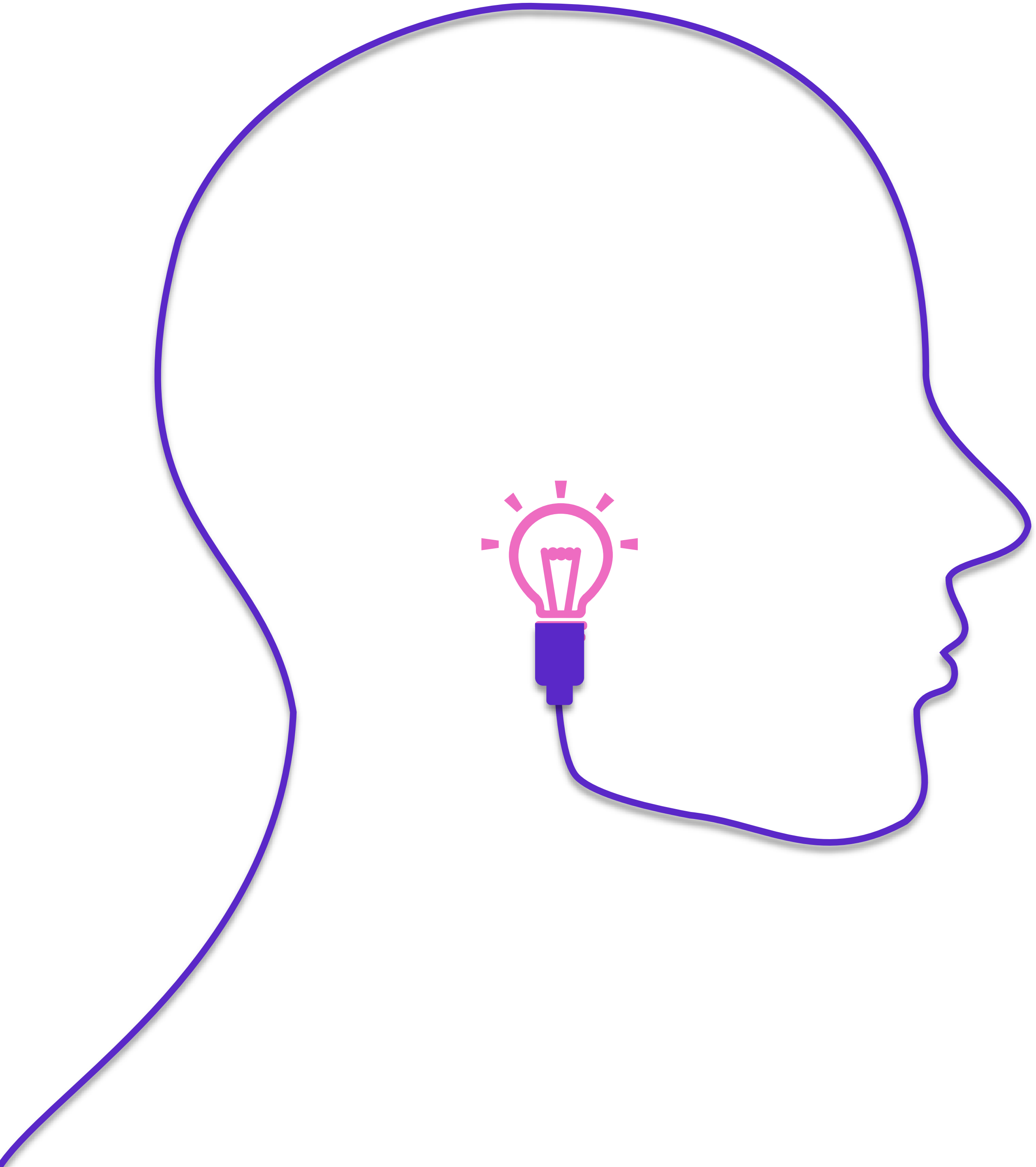
Aditya Aggarwal
IIT2019210

Ambika Singh
IIB2019017

Avneesh Kumar
IIB2019010

Divy Agarwal
IIT2019211

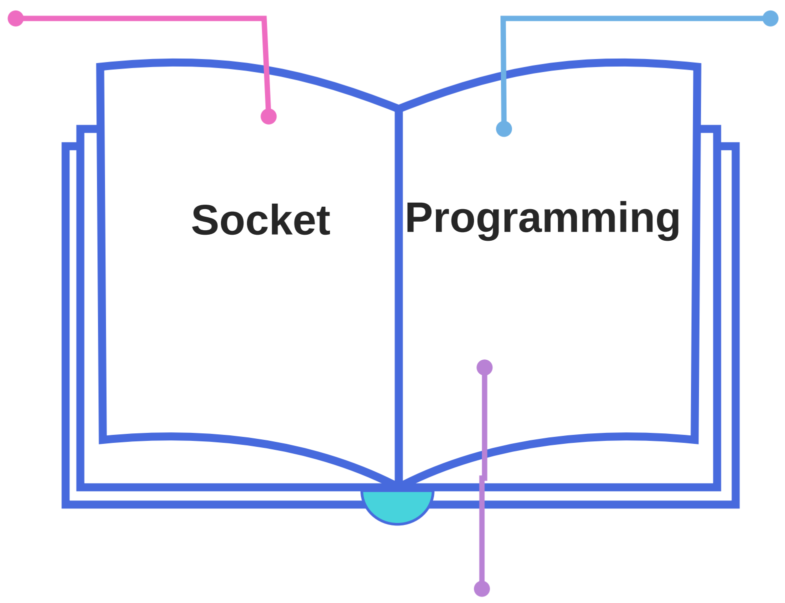
Table of content



1. Basic Concepts

Socket Programming

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

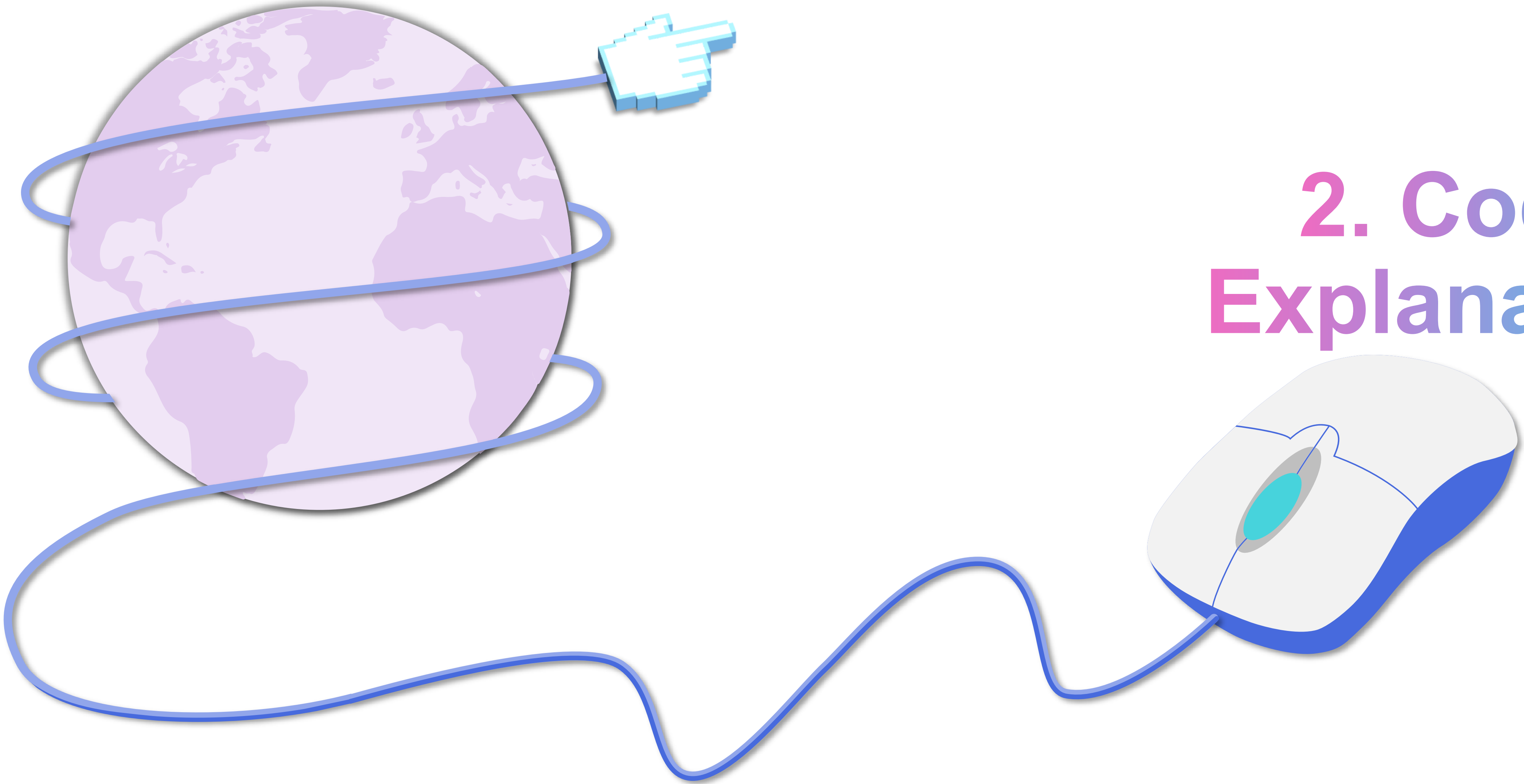


Client Process

This is the process, which typically makes a request for information. After getting the response, this process may terminate or may do some other processing.

Server Process

This is the process which takes a request from the clients. After getting a request from the client, this process will perform the required processing, gather the requested information, and send it to the requestor client. Once done, it becomes ready to serve another client. Server processes are always alert and ready to serve incoming requests.



2. Code & Explanation

Server.c

This function takes 2 integers and return their gcd.

This function takes an array as input and returns the lcm of fixed number of integers present in the array

```
int gcd(int a, int b){
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
int lcm(int arg[]){
    int lcm=arg[0];
    for(int i=1;i<4;i++){
        lcm=((arg[i]*lcm)/gcd(arg[i],lcm));
    }
}
```

This if condition will ensure that port number is provided or not. If port is not provided then server process can't be started rather it will display the error and exits.

These are the variable declarations used in the server process.

Structure describing an Internet socket address.

Created a unsigned int of `socklen_t` type

```
int main(int argc,char * argv[]){
    if(argc<2){
        fprintf(stderr, "Please provide the port number");
        exit(1);
    }
    int sockfd, newsockfd, portno, n, ans=1;
    char buffer[1024];
    pid_t childpid;
    struct sockaddr_in server_address,client_address;
    socklen_t client_len;
```

Create a new socket of TYPE in DOMAIN(here is AF_INET), using PROTOCOL. If PROTOCOL is zero, one is chosen automatically(protocol is 0 so that it can choose by itself). Returns a file descriptor for the new socket, or -1 for errors.

- Sets N(here is sizeof(server_address)) bytes of S(here is server address) to 0.
- Port number that is stored at first index of argv is converted into integer by atoi and then stored in portno

The elements in the structure are assigned their values.

This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as: “address already in use”.

After creation of the socket, bind function binds the socket to the address and port number specified in addr(custom data structure).

```
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd<0){
        perror("Error in creating a socket");
        exit(1);
    }
    bzero((char *) &server_address,sizeof(server_address));
    portno=atoi(argv[1]);
    server_address.sin_family=AF_INET;
    server_address.sin_addr.s_addr=INADDR_ANY;
    server_address.sin_port=htons(portno);
    int tr=1;
    if(setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&tr,sizeof
(int))==-1){
        perror("Not a useable port");
        exit(1);
    }
    if(bind(sockfd,(struct sockaddr *) &server_address,size
of(server_address))<0){
        perror("Not able to bind");
        exit(1);
    }
```

`listen(sockfd,5)` Prepares to accept connections on socket FD. 5 connection requests will be queued before further requests are refused.

`Stores the size of client_address to client_len.`

```
listen(sockfd,5);
    client_len=sizeof(client_address);
    while(1){
        newsockfd = accept(sockfd, (struct sockaddr*)&client_address,
 &client_len);
        if(newsockfd < 0){
            exit(1);
        }
        printf("Connection accepted from %s:%d\n", inet_ntoa(client_a
ddress.sin_addr), ntohs(client_address.sin_port));
```

Awaits for a connection on socket FD.
When a connection arrives, open a new socket to communicate with it, set *ADDR (which is client_len bytes long) to the address of the connecting peer and *ADDR_LEN to the address's actual length.
Returns the new socket's descriptor, or -1 for errors, on error will print it and exits

fork: It creates a child process and this block will run for child only.
close: child sockfd is closed.
bzero: Sets 1024 bytes of buffer to 0.
read: Reads N BYTES into BUF(arg) from FD(newsockfd). Return the number read, -1 for errors or 0 for EOF.
lcm: calculates and stores the result.
write: Writes 4 bytes of ans to newsockfd. Return the number written, or -1.

At the end close both the FD.

```
if((childpid = fork()) == 0){
      close(sockfd);
      bzero(buffer,1024);
      printf("\tNow taking input from client\n");
      int arg[4];
      n=read(newsockfd,arg,sizeof(arg));
      if(n<0){
        perror("Error while reading");
        exit(1);
      }
      ans=lcm(arg);
      n=write(newsockfd,&ans,sizeof(int));
      if(n<0){
        perror("Error while writing");
        exit(1);
      }
    }

  }
  close(sockfd);
  close(newsockfd);
```

client.c

Structures describing an Internet socket address. These are the variable declarations used in the client process.

Port number that is stored at second index of *argv* is converted into integer by `atoi` and then stored in `portno`.

Creates a socket as in server and also checks for errors

Return entry from host data base for host with NAME(argv[1]).

`bzero` Sets the given amount bytes of server address to 0 as in server process.
Structure has been assigned the value of `sin_family` **as AF_INET.**

```
int sockfd,portno,n;
struct sockaddr_in server_address;
struct hostent *server;
char buffer[255];
if(argc<3){
    fprintf(stderr,"usage %s hostname port\n",argv[0]);
    exit(1);
}
portno=atoi(argv[2]);
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0){
    perror("Error in creating a socket");
    exit(1);
}
server=gethostbyname(argv[1]);
if(server==NULL){
    fprintf(stderr,"Error, no such host");
    exit(1);
}
bzero((char *) &server_address,sizeof(server_address));
server_address.sin_family=AF_INET;
```

Copy the given bytes of `server->h_addr_list[0]` to `server_address.sin_addr.s_addr.`

Structure has been assigned the value of port number

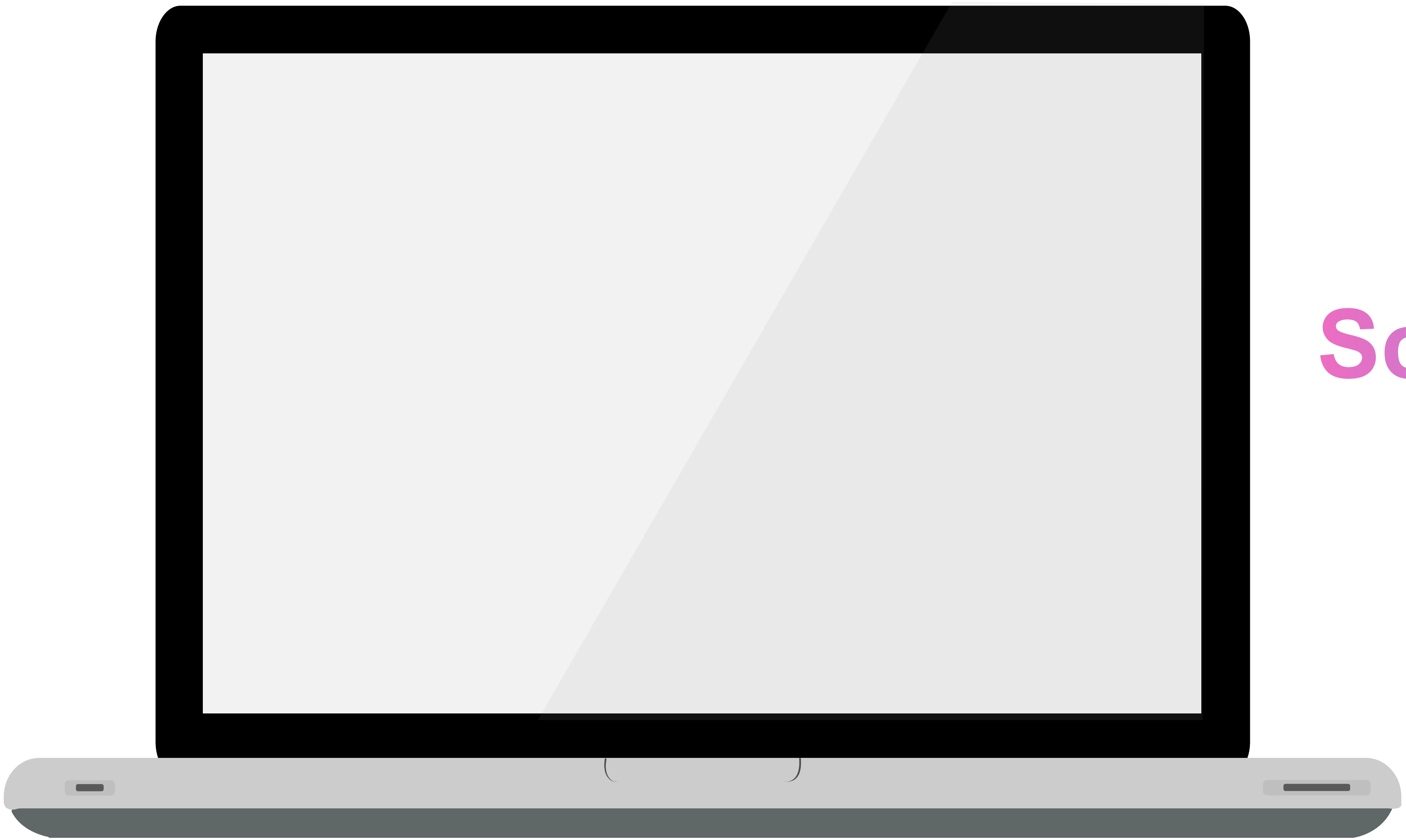
The `connect()` system call connects the socket referred to by the file descriptor `sockfd` to the address specified by `server_address`. Server’s address and port is specified in addr.

Since currently we can calculate lcm of 4 integers so 4 integers are being scanned by the user

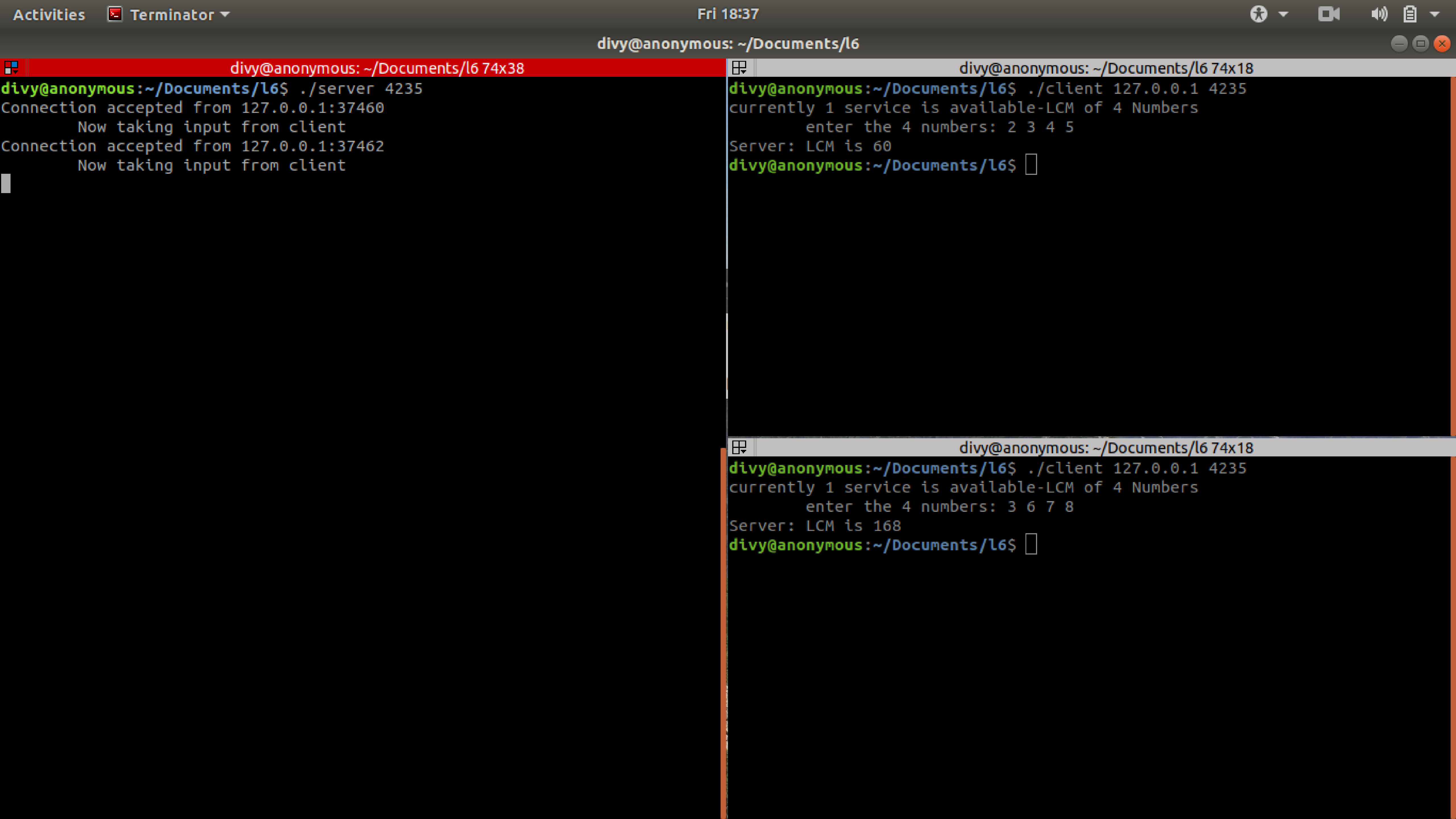
Reads the answer from the socket, if there is any error then prints it and exit.

Prints the value of lcm that had received from server through socket connection.

```
    bcopy((char *) server-
>h_addr_list[0],&server_address.sin_addr.s_addr,server-
>h_length);
    server_address.sin_port=htons(portno);
    if(connect(sockfd,(struct sockaddr *) &server_addre
ss,sizeof(server_address))<0){
        perror("Connection Failed");
        exit(1);
    }
    printf("currently 1 service is available-
LCM of 4 Numbers\n");
    printf("\tenter the 4 numbers: ");
    int arg[4];
    for(int i=0;i<4;i++){
        scanf("%d",&arg[i]);
    }
    n=write(sockfd,arg,sizeof(arg));
    if(n<0){
        perror("Error while writing");
        exit(1);
    }
    int ans;
    n=read(sockfd,&ans,sizeof(int));
    if(n<0){
        perror("Error while reading");
        exit(1);
    }
    printf("Server: LCM is %d\n",ans);
```



3. Output Screenshots



Activities Terminator Fri 18:37

divy@anonymous: ~/Documents/l6

divy@anonymous: ~/Documents/l6 74x38

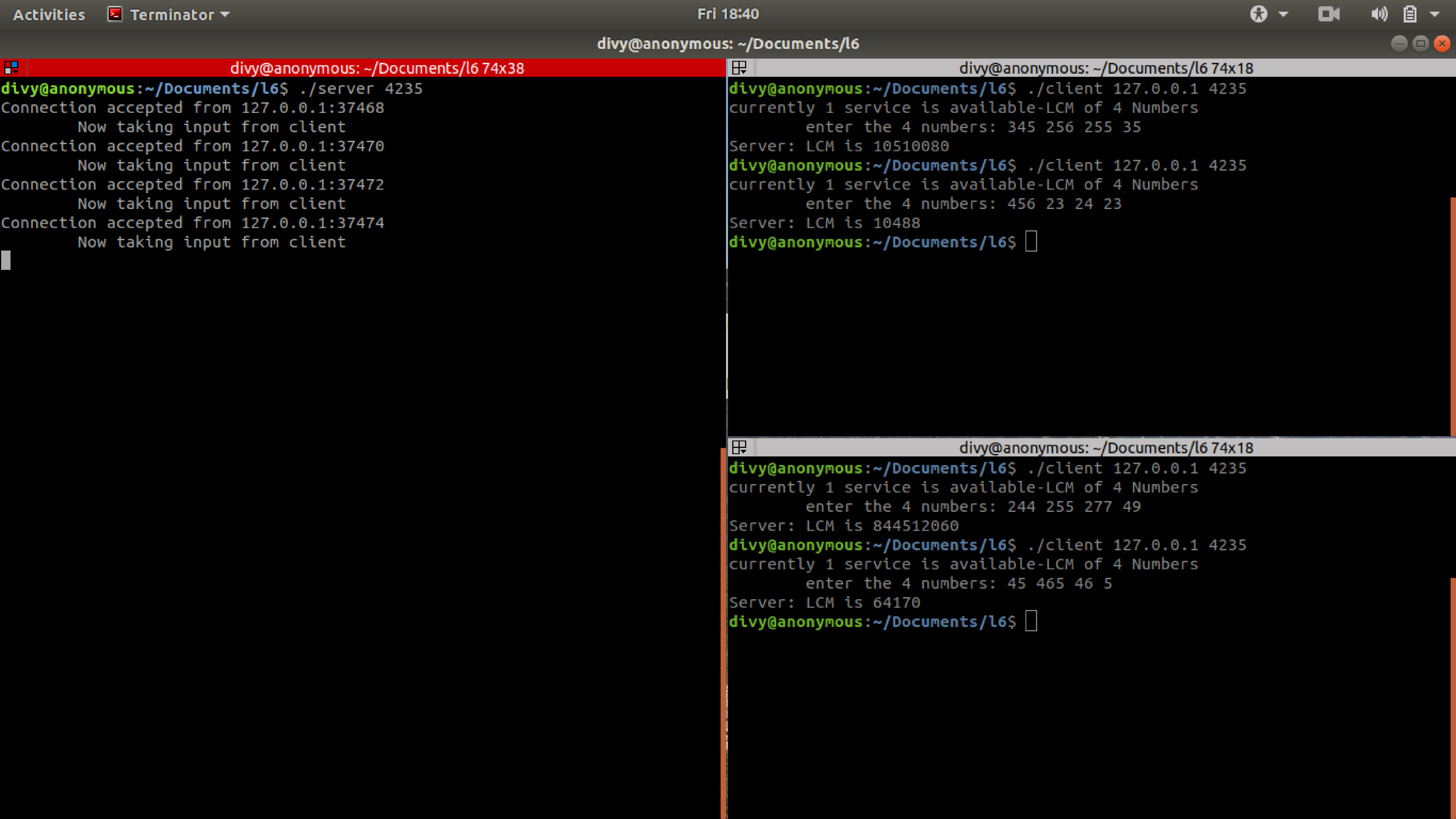
```
divy@anonymous:~/Documents/l6$ ./server 4235
Connection accepted from 127.0.0.1:37460
        Now taking input from client
Connection accepted from 127.0.0.1:37462
        Now taking input from client
```

divy@anonymous: ~/Documents/l6 74x18

```
divy@anonymous:~/Documents/l6$ ./client 127.0.0.1 4235
currently 1 service is available-LCM of 4 Numbers
        enter the 4 numbers: 2 3 4 5
Server: LCM is 60
divy@anonymous:~/Documents/l6$ 
```

divy@anonymous: ~/Documents/l6 74x18

```
divy@anonymous:~/Documents/l6$ ./client 127.0.0.1 4235
currently 1 service is available-LCM of 4 Numbers
        enter the 4 numbers: 3 6 7 8
Server: LCM is 168
divy@anonymous:~/Documents/l6$ 
```

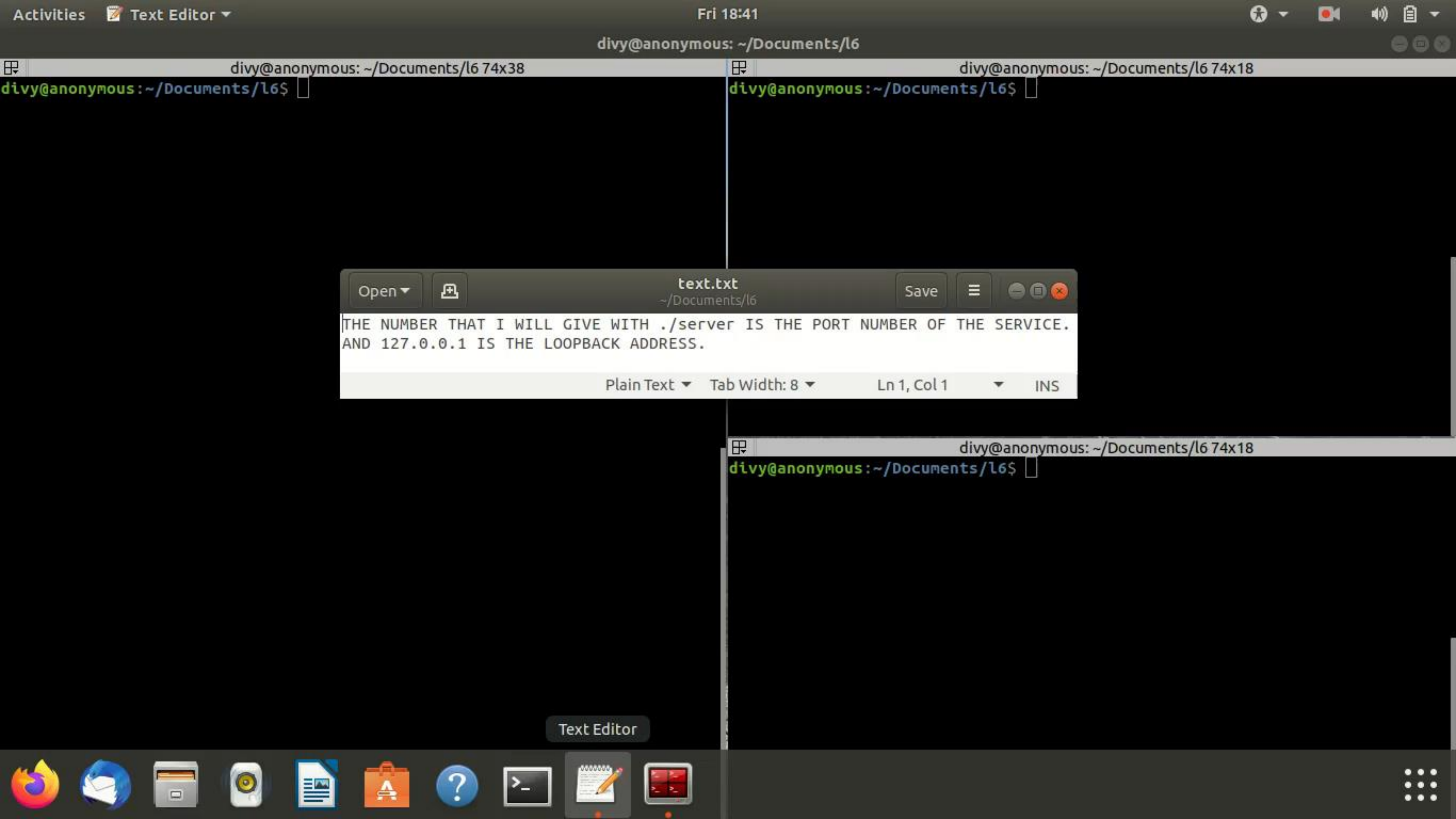


```
divy@anonymous:~/Documents/l6$ ./server 4235
Connection accepted from 127.0.0.1:37468
        Now taking input from client
Connection accepted from 127.0.0.1:37470
        Now taking input from client
Connection accepted from 127.0.0.1:37472
        Now taking input from client
Connection accepted from 127.0.0.1:37474
        Now taking input from client
```

```
divy@anonymous:~/Documents/l6$ ./client 127.0.0.1 4235
currently 1 service is available-LCM of 4 Numbers
        enter the 4 numbers: 345 256 255 35
Server: LCM is 10510080
divy@anonymous:~/Documents/l6$ ./client 127.0.0.1 4235
currently 1 service is available-LCM of 4 Numbers
        enter the 4 numbers: 456 23 24 23
Server: LCM is 10488
divy@anonymous:~/Documents/l6$
```

```
divy@anonymous:~/Documents/l6$ ./client 127.0.0.1 4235
currently 1 service is available-LCM of 4 Numbers
        enter the 4 numbers: 244 255 277 49
Server: LCM is 844512060
divy@anonymous:~/Documents/l6$ ./client 127.0.0.1 4235
currently 1 service is available-LCM of 4 Numbers
        enter the 4 numbers: 45 465 46 5
Server: LCM is 64170
divy@anonymous:~/Documents/l6$
```

Activities
Terminator
Fri 18:40
divy@anonymous: ~/Documents/l6
divy@anonymous: ~/Documents/l6 74x38
divy@anonymous: ~/Documents/l6 74x18
divy@anonymous: ~/Documents/l6 74x18


Activities
Text Editor
Fri 18:41
divy@anonymous: ~/Documents/l6
divy@anonymous: ~/Documents/l6 74x38
divy@anonymous: ~/Documents/l6 74x18
divy@anonymous: ~/Documents/l6 74x18
Open
text.txt
~/Documents/l6
Save
Plain Text
Tab Width: 8
Ln 1, Col 1
INS
Text Editor

```
divy@anonymous:~/Documents/l6$
```

```
divy@anonymous:~/Documents/l6$
```

```
divy@anonymous:~/Documents/l6$
```

```
THE NUMBER THAT I WILL GIVE WITH ./server IS THE PORT NUMBER OF THE SERVICE.
AND 127.0.0.1 IS THE LOOPBACK ADDRESS.
```

THANK YOU