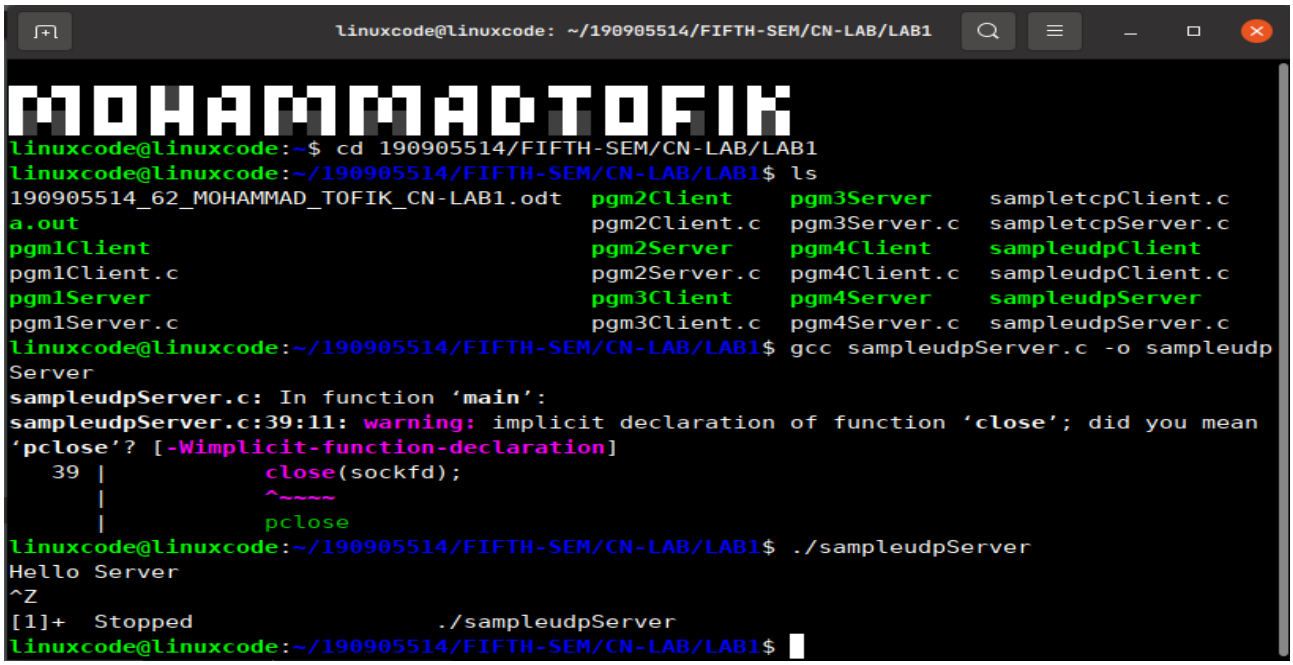
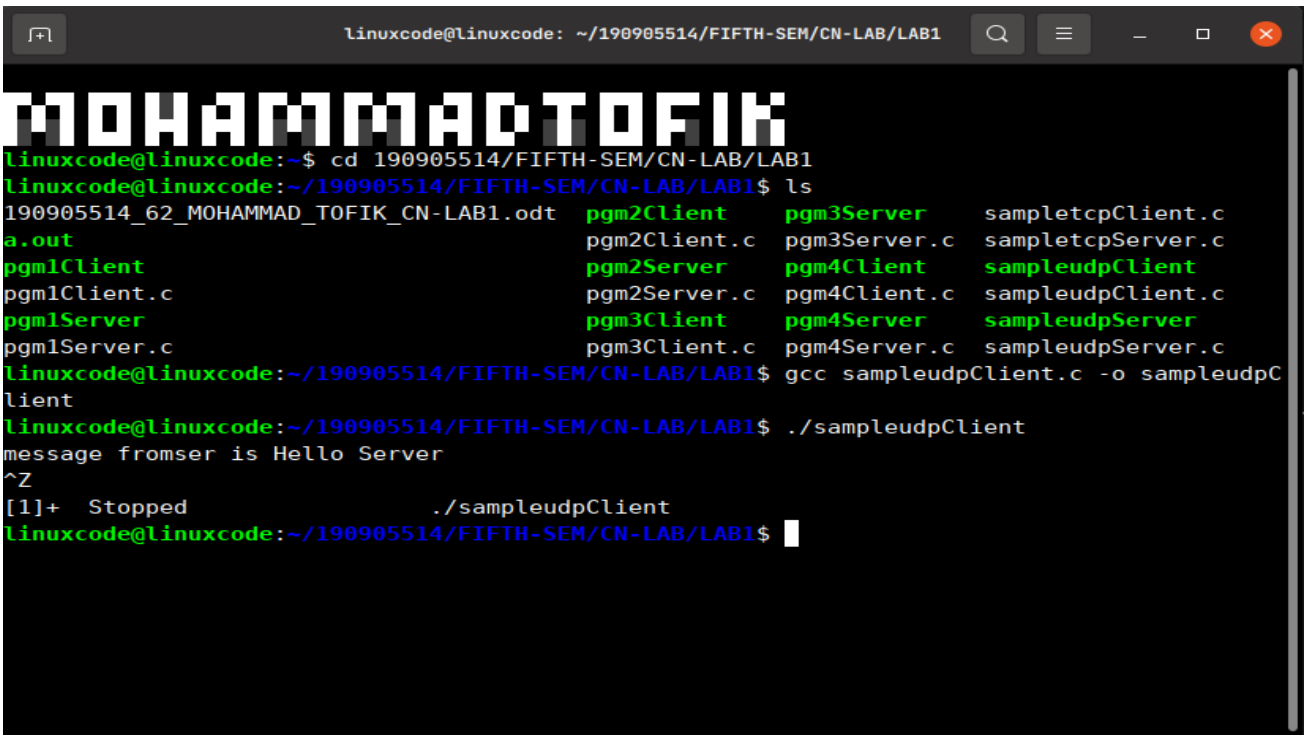
190905514 MOHAMMAD TOFIK C SECTION
Roll No : 62 FIFTH-SEM

[LAB-SESSION-1]
CN-LAB-1

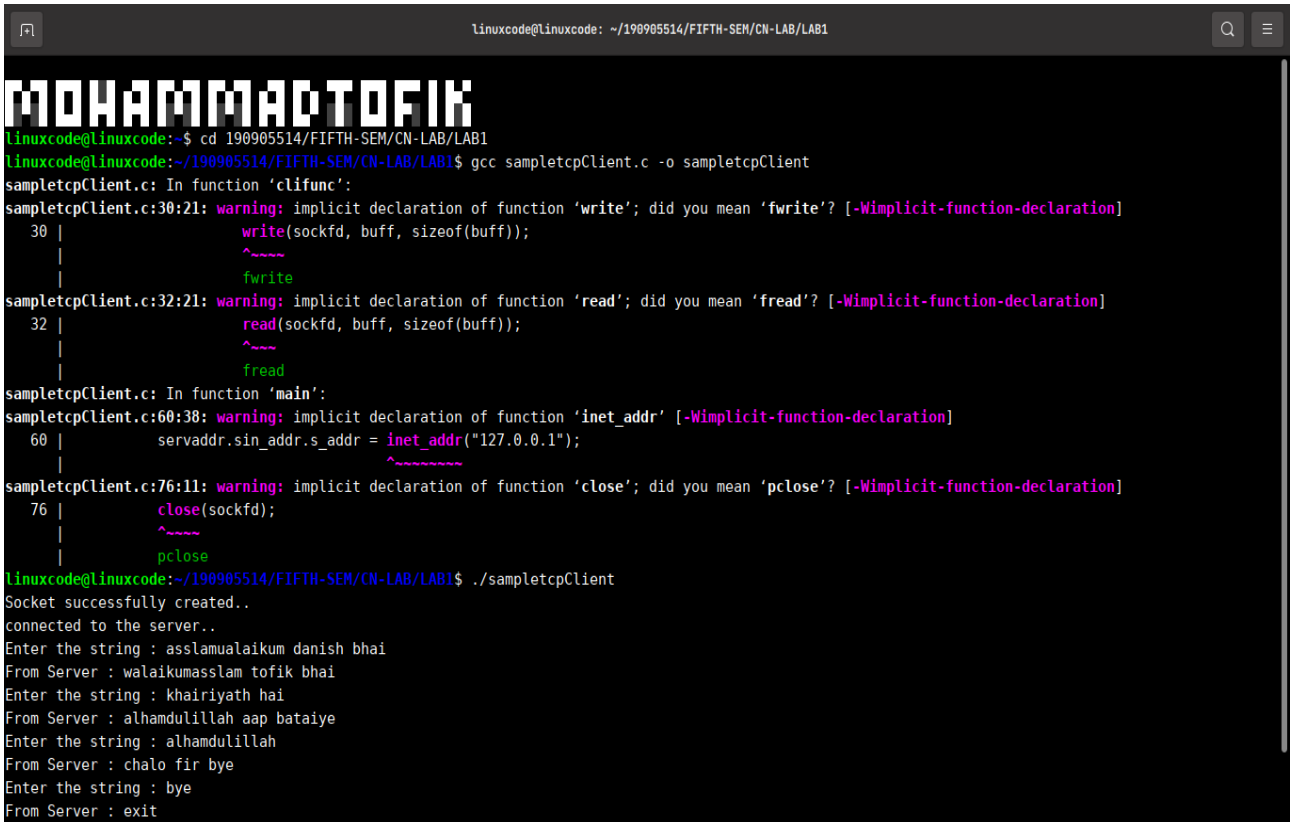
sample program output :

sampleudpServer.c
sampleudpClient.c





sampletcpServer.c
sampletcpClient.c



EXCERCISE :

1. Write a UDP client-server program where client sends rows of a matrix to the server combines them together as a two dimensional matrix and display the same.

pgm1udpClient.c

```c
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <stdlib.h>
#define PORT 5000
#define MAXLINE 1000
void main()
{
int buffer[100];
int sockfd, n, len;
struct sockaddr_in servaddr, cliaddr;
// using a square matrix of 3*2
printf("Enter the elements of the first row\n");
int a, b, c;
scanf("%d %d %d", &a, &b, &c);
printf("Enter the elements of the second row \n");
int d, e, f;
scanf("%d%d%d", &d, &e, &f);
// clear servaddr
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);
servaddr.sin_family = AF_INET;
int message[6];
message[0] = a;
message[1] = b;
message[2] = c;
message[3] = d;
```

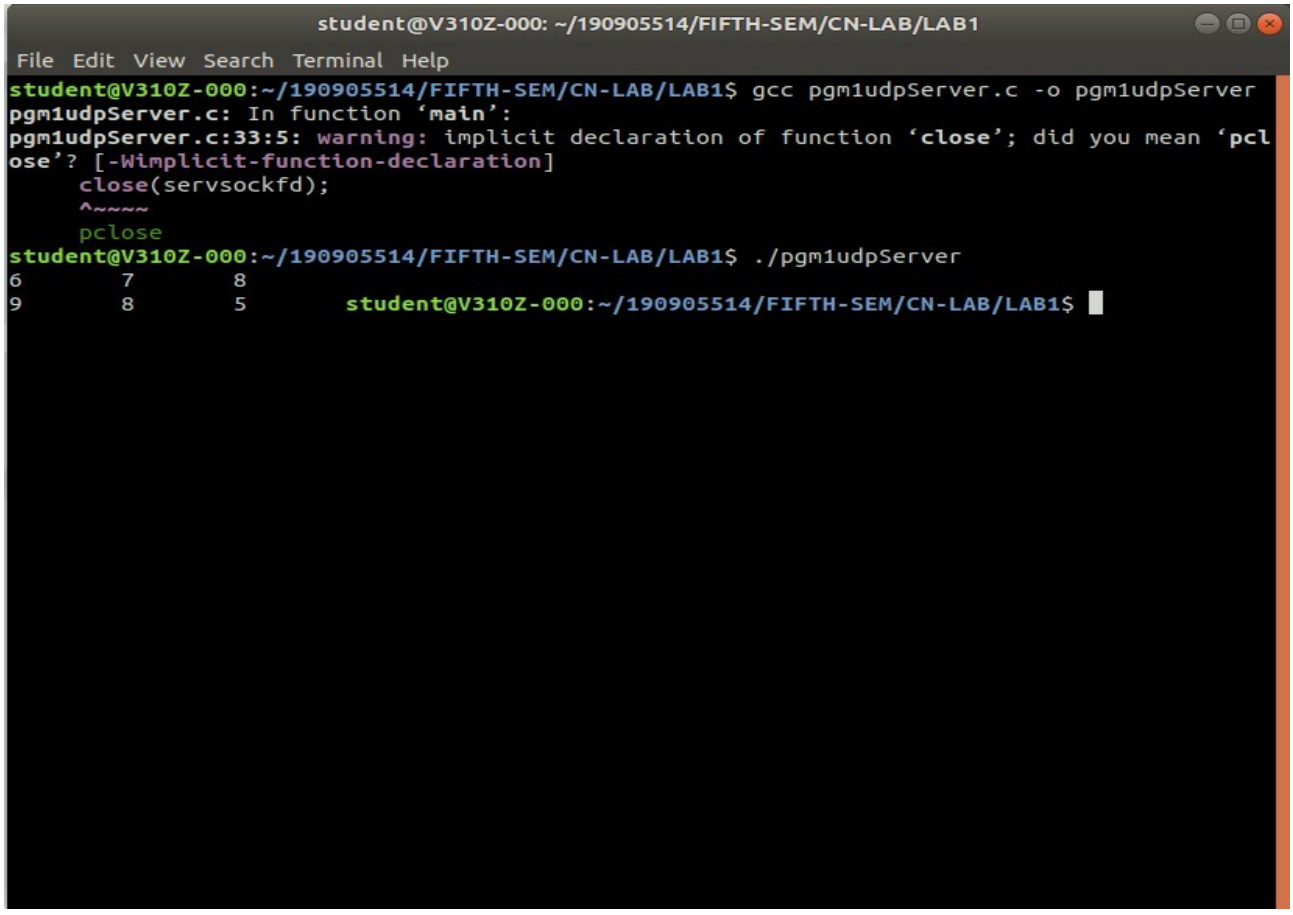
```
message[4] = e;
message[5] = f;
// create datagram socket
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
sendto(sockfd, message, MAXLINE, 0, (struct sockaddr *)&servaddr,
sizeof(servaddr));
len = sizeof(cliaddr);
// waiting for response
n = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr *)&cliaddr,
&len);
buffer[n] = '\0';
printf("Message from Server is \n");
// Just chck if it gives correct outpur or not , connection is alreadyesablished
for (int i = 0; i < 6; i++)
{
// hardcoded till 3
printf("The %d th element of the matrix is :-%d \n", i, (buffer[i]));
} // close the descriptor
close(sockfd);
}
```

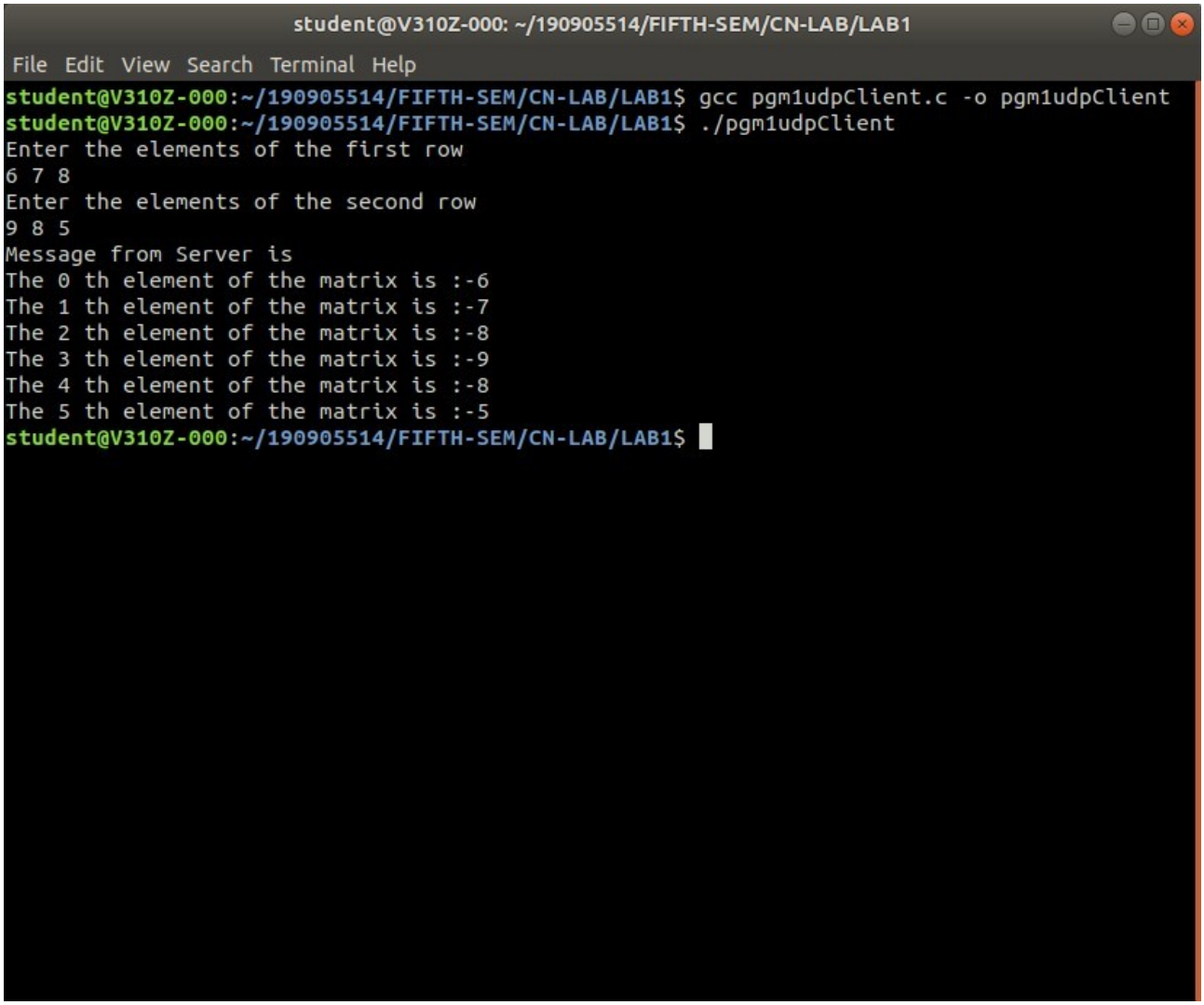
pgm1udpServer.c

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 5000
#define MAXLINE 1000
void main()
{
int buffer[100];
int servsockfd, i, len, n;
struct sockaddr_in servaddr, cliaddr;
bzero(&servaddr, sizeof(servaddr));
// Create a UDP Socket
servsockfd = socket(AF_INET, SOCK_DGRAM, 0);
```

```
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);
servaddr.sin_family = AF_INET;
// bind server address to socket descriptor
bind(servsockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
// receive the datagram
len = sizeof(cliaddr);
n = recvfrom(servsockfd, buffer, sizeof(buffer), 0, (struct sockaddr *)&cliaddr,
&len);
for (i = 0; i < 3; i++)
printf("%d\t", buffer[i]);
printf("\n");
for (i = 3; i < 6; i++)
printf("%d\t", buffer[i]);
// Echoing back to the client
sendto(servsockfd, buffer, n, 0, (struct sockaddr *)&cliaddr, sizeof(cliaddr));
// close the descriptor
close(servsockfd);
}
```

OUTPUT :





2. Write a TCP client which sends a string to a server program. Server displays the string along with client IP and ephemeral port number. Server then responds to the client by echoing back the string in uppercase. The process continues until one of them types “QUIT”.

pgm2tcpClient.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <ctype.h>
```

```c
#include <string.h>
#define MAX 50
void clifunc(int sockfd)
{
char buff[MAX];
int n = 0, recv_len = 0;
while (1)
{
memset(buff, 0, sizeof(buff));
printf("Type message\n");
//while((buff[n++]=getchar())!='\n');
scanf("%s", buff);
write(sockfd, buff, sizeof(buff));
if (strcmp("quit", buff) == 0)
{
printf("client closing\n");
return;
}
memset(buff, 0, sizeof(buff));
n = read(sockfd, buff, sizeof(buff));
buff[n] = '\n';
printf("SERVER : %s\n", buff);
}
}
int main(int argc, char const *argv[])
{
int sockfd;
int len;
struct sockaddr_in server_address;
int result;
char ch;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = htons(7280);
len = sizeof(server_address);
result = connect(sockfd, (struct sockaddr *)&server_address, len);
if (result == -1)
```

```
{
printf("connection error\n");
exit(0);
}
clifunc(sockfd);
close(sockfd);
return 0;
}
```

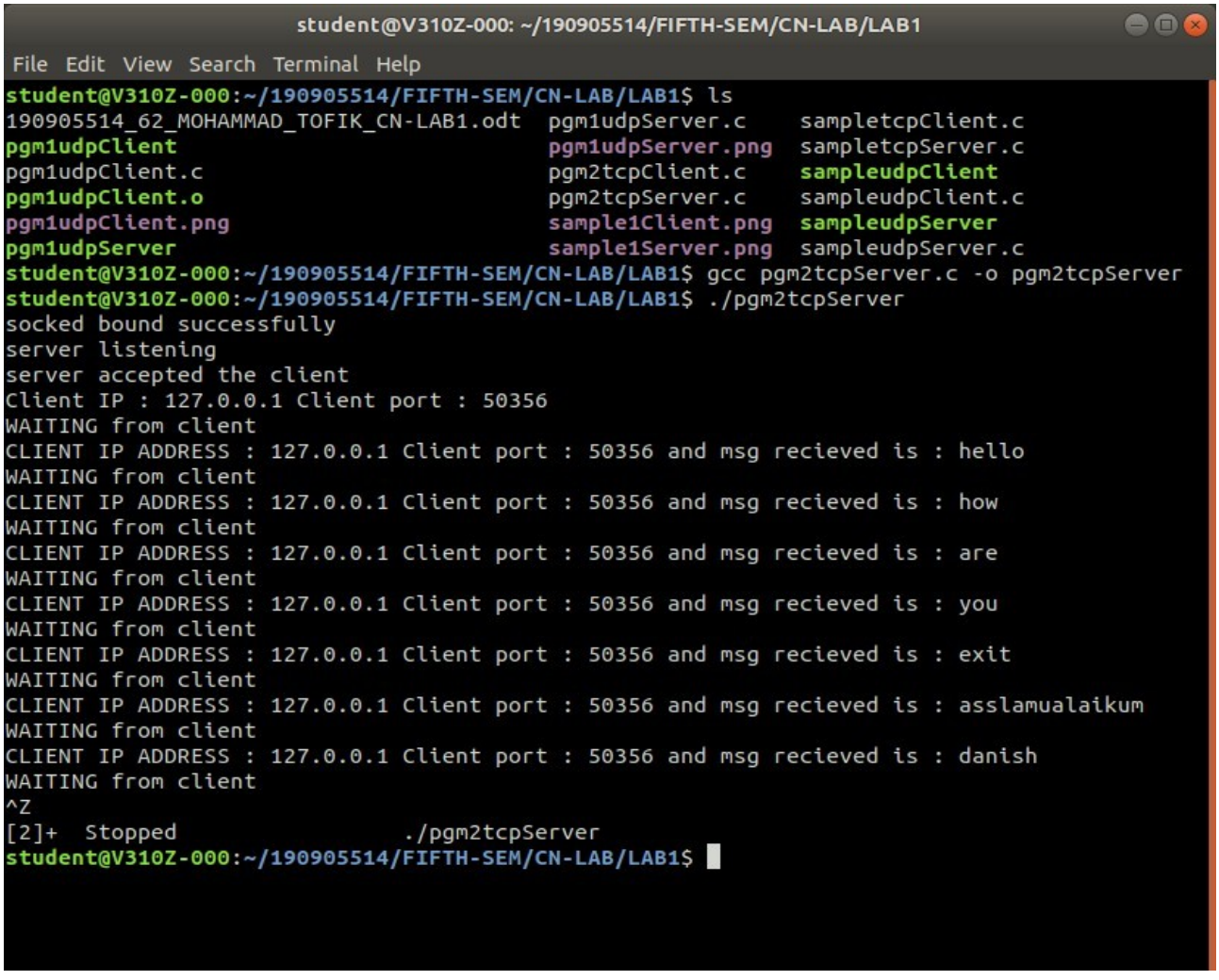
pgm2tcpServer.c

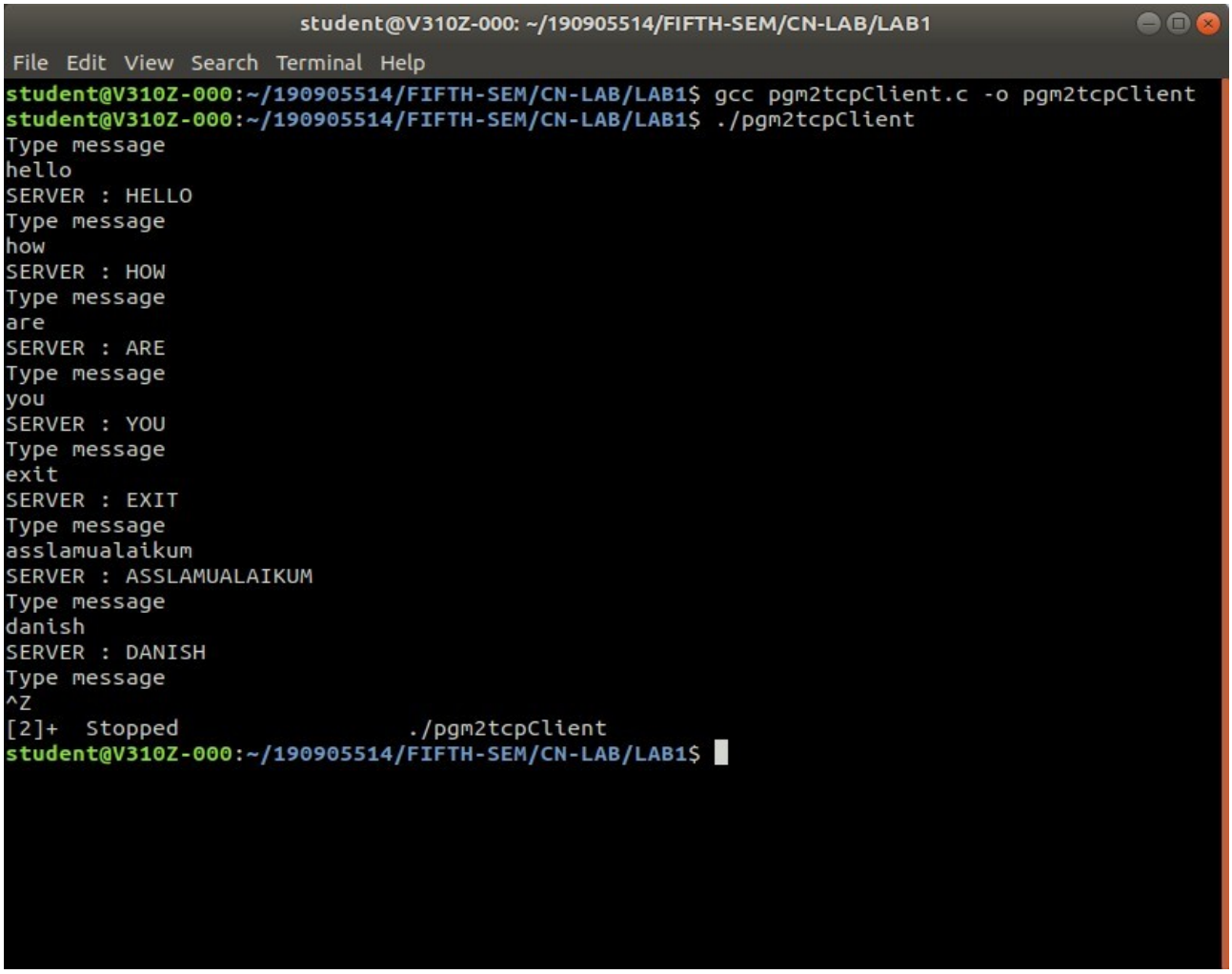
```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#define MAX 50
void servfunc(int conn_fd, struct sockaddr_in client_address)
{
char buff[MAX];
int n = 0;
char *ip_add = inet_ntoa(client_address.sin_addr);
int port = client_address.sin_port;
printf("Client IP : %s Client port : %d \n", ip_add, port);
while (1)
{
printf("WAITING from client\n");
memset(buff, 0, sizeof(buff));
n = read(conn_fd, buff, sizeof(buff));
buff[n] = '\n';
printf("CLIENT IP ADDRESS : %s Client port : %d and msg recieved is : %s \
n",
ip_add, port, buff);
if (strcmp("quit", buff) == 0)
{
```

```
printf("server is getting closed...\n");
return;
}
for (int i = 0; i < n; i++)
{
buff[i] = toupper(buff[i]);
}
write(conn_fd, buff, sizeof(buff));
}
}
int main()
{
int server_sockfd, conn_sockfd;
int server_len, client_len;
struct sockaddr_in server_address;
struct sockaddr_in client_address;
server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = htons(7280);
server_len = sizeof(server_address);
if (bind(server_sockfd, (struct sockaddr *)&server_address, server_len) != 0)
{
printf("socket binding failed\n");
exit(0);
}
else
{
printf("socked bound successfully\n");
}
if (listen(server_sockfd, 2) != 0)
{
printf("listen failed\n");
exit(0);
}
else
{
printf("server listening\n");
```

```
}
client_len = sizeof(client_address);
conn_sockfd = accept(server_sockfd, (struct sockaddr *)&client_address,
&client_len);
if (conn_sockfd < 0)
{
printf("server accept failed\n");
exit(0);
}
else
{
printf("server accepted the client\n");
}
servfunc(conn_sockfd, client_address);
close(server_sockfd);
return 0;
}
```

OUTPUT :





3.DayTime server : where client sends request to time server to send current time.Server responds by sending the current time.
[Hint : read man pages of asctime() and localtime()].Display server process id at the client side along with time.

pgm3Client.c

```c
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
```

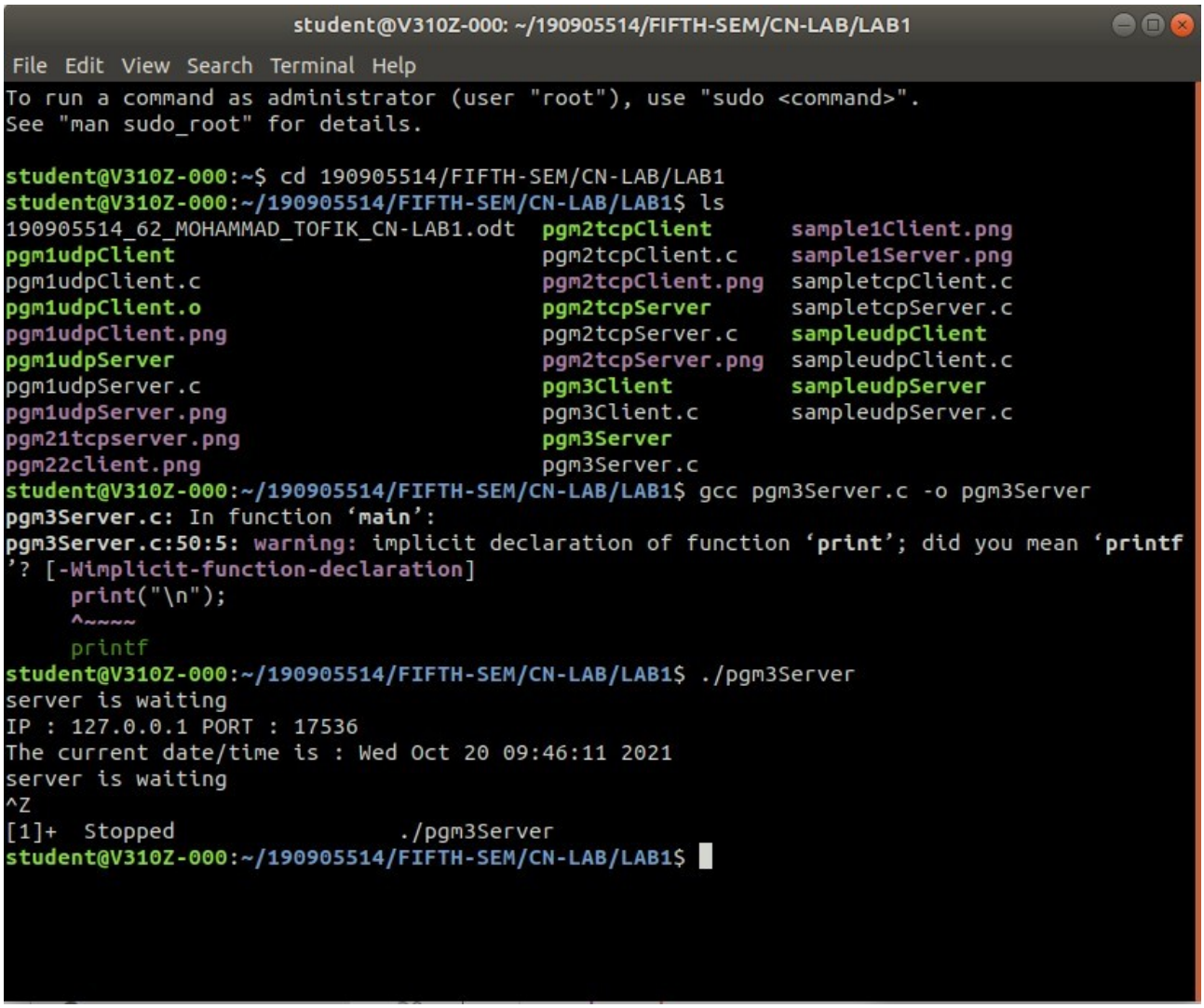
```c
#include <arpa/inet.h>
int main()
{
int sockfd;
int len;
struct sockaddr_in address;
struct tm *timeinfo;
int result;
char *reply;
int hour, mins, sec, pid;
/* Create a socket for the client. */
sockfd = socket(AF_INET, SOCK_STREAM, 0);
/* Name the socket, as agreed with the server. */
address.sin_family = AF_INET;
address.sin_addr.s_addr = inet_addr("127.0.0.1");
address.sin_port = 9734;
len = sizeof(address);
/* Now connect our socket to the server's socket. */
result = connect(sockfd, (struct sockaddr *)&address, len);
if (result == -1)
{
perror("oops: client2");
exit(1);
}
/* We can now read/write via sockfd. */
printf(" Sending request to get the time\n");
read(sockfd, &hour, 1);
read(sockfd, &mins, 1);
read(sockfd, &sec, 1);
read(sockfd, &pid, 1);
printf("%d:%d:%d", hour, mins, sec);
printf(" The process id is: %d", pid);
close(sockfd);
exit(0);
return 0;
}
```

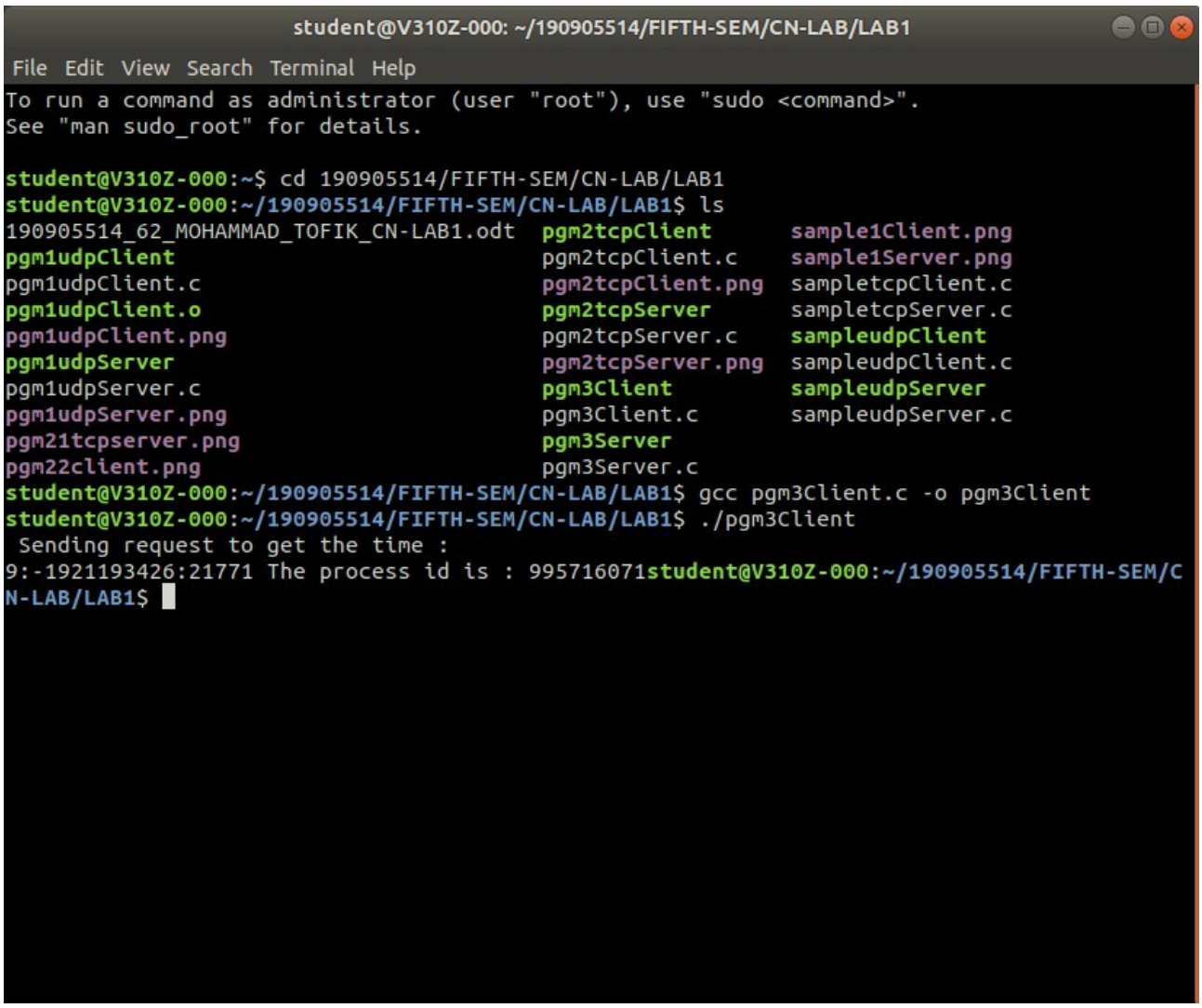
pgm3Server.c

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>

int main()
{
time_t rawtime;
struct tm *timeinfo;
char *reply;
int server_sockfd, client_sockfd;
int server_len, client_len;
struct sockaddr_in server_address;
struct sockaddr_in client_address;
int hour, mins, sec, pid;

/* Create an unnamed socket for the server. */

server_sockfd = socket(AF_INET, SOCK_STREAM, 0);

/* Name the socket. */

server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = 9734;
server_len = sizeof(server_address);
bind(server_sockfd, (struct sockaddr *)&server_address, server_len);

/* Create a connection queue and wait for clients. */

listen(server_sockfd, 5);
while (1)
{
```

```
char ch;

printf("server waiting\n");

/* Accept a connection. */

client_len = sizeof(client_address);
client_sockfd = accept(server_sockfd, (struct sockaddr *)&client_address,
&client_len);

/* We can now read/write to client on client_sockfd. */
//char *inet_ntoa(client_addr.sin_addr);

char *ip_add = inet_ntoa(client_address.sin_addr);
int port = client_address.sin_port;

printf("IP:%s PORT:%d\n", ip_add, port);

//get the time

time(&rawtime);
timeinfo = localtime(&rawtime);
reply = asctime(timeinfo);
printf("The current date/time is: %s", reply);
hour = timeinfo->tm_hour;
mins = timeinfo->tm_min;
sec = timeinfo->tm_sec;
pid = getpid();
write(client_sockfd, &hour, 1);
write(client_sockfd, &mins, 1);
write(client_sockfd, &sec, 1);
write(client_sockfd, &pid, 1);
//close(client_sockfd);
}
return 0;
}
```

OUTPUT :


student@V310Z-000: ~/190905514/FIFTH-SEM/CN-LAB/LAB1
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
student@V310Z-000:~$ cd 190905514/FIFTH-SEM/CN-LAB/LAB1
student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$ ls
190905514_62_MOHAMMAD_TOFIK_CN-LAB1.odt
pgm1udpClient
pgm1udpClient.c
pgm1udpClient.o
pgm1udpClient.png
pgm1udpServer
pgm1udpServer.c
pgm1udpServer.png
pgm21tcpserver.png
pgm22client.png
pgm2tcpClient
pgm2tcpClient.c
pgm2tcpClient.png
pgm2tcpServer
pgm2tcpServer.c
pgm2tcpServer.png
pgm3Client
pgm3Client.c
pgm3Server
pgm3Server.c
sample1Client.png
sample1Server.png
sampletcpClient.c
sampletcpServer.c
sampleudpClient
sampleudpClient.c
sampleudpServer
sampleudpServer.c
student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$ gcc pgm3Server.c -o pgm3Server
pgm3Server.c: In function 'main':
pgm3Server.c:50:5: warning: implicit declaration of function 'print'; did you mean 'printf'? [-Wimplicit-function-declaration]
print("\n");
^~~~~
printf
student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$ ./pgm3Server
server is waiting
IP : 127.0.0.1 PORT : 17536
The current date/time is : Wed Oct 20 09:46:11 2021
server is waiting
^Z
[1]+ Stopped ./pgm3Server
student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$


student@V310Z-000: ~/190905514/FIFTH-SEM/CN-LAB/LAB1
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
student@V310Z-000:~$ cd 190905514/FIFTH-SEM/CN-LAB/LAB1
student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$ ls
190905514_62_MOHAMMAD_TOFIK_CN-LAB1.odt
pgm1udpClient
pgm1udpClient.c
pgm1udpClient.o
pgm1udpClient.png
pgm1udpServer
pgm1udpServer.c
pgm1udpServer.png
pgm21tcpserver.png
pgm22client.png
pgm2tcpClient
pgm2tcpClient.c
pgm2tcpClient.png
pgm2tcpServer
pgm2tcpServer.c
pgm2tcpServer.png
pgm3Client
pgm3Client.c
pgm3Server
pgm3Server.c
sample1Client.png
sample1Server.png
sampletcpClient.c
sampletcpServer.c
sampleudpClient
sampleudpClient.c
sampleudpServer
sampleudpServer.c
student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$ gcc pgm3Client.c -o pgm3Client
student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$ ./pgm3Client
Sending request to get the time :
9:-1921193426:21771 The process id is : 995716071student@V310Z-000:~/190905514/FIFTH-SEM/CN-LAB/LAB1$