NETWORK PROGRAMMING

CSX-354

LAB PRACTICALS RECORD

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr. B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR – 144011, PUNJAB (INDIA)

Submitted To:

Mr. MANOJ KUMAR
Asst. Professor
Department of CSE

Submitted By:

Nikhil Bansal
13103011
6th Semester

INDEX

PROGRAM-1
Simple TCP Communication

This Program illustrates communication between client and server using TCP Protocol.

Server Program:

```
#include<sys/types.h>
#include<sys/socket.h>
#include<string.h>
#include<stdio.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
        struct sockaddr_in servaddr;
        char output[20];
        int n, temp, i, j, mysockfd, clientfd;
        struct sockaddr_in client;
        int clilen=sizeof(client);

        // create socket at server
        mysockfd = socket(AF_INET, SOCK_STREAM, 0);
        if(mysockfd<0)
        {
                perror("Socket failed");
                return -1;
        }

        // create the server address
        memset(&servaddr, 0, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(5000);

        // bind the server address to the socket
        temp = bind(mysockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
        if(temp<0)
        {
                perror("Bind failed");
```

```
		return -1;
	}
	else
	{
		printf("Bind successful\n");
	}

	// listen to the requests with at max 2 requests
	temp = listen(mysockfd, 2);
	if(temp<0)
	{
		perror("Listen failed");
		return -1;
	}
	else
	{
		printf("Listen successful\n");
	}

	// accept the clients request
	clientfd = accept(mysockfd, (struct sockaddr *)&client, (unsigned int *)&clilen);
	if(clientfd<0)
	{
		perror("Accept failed");
		return -1;
	}
	else
	{
		printf("Accept successful\n");
	}

	// communicate with client
	while(1)
	{
		if((n=read(clientfd,output,20-1))==0)
			break;
		if(n>0)
		{
			output[n]='\0';
			printf("%s\n",output);
```

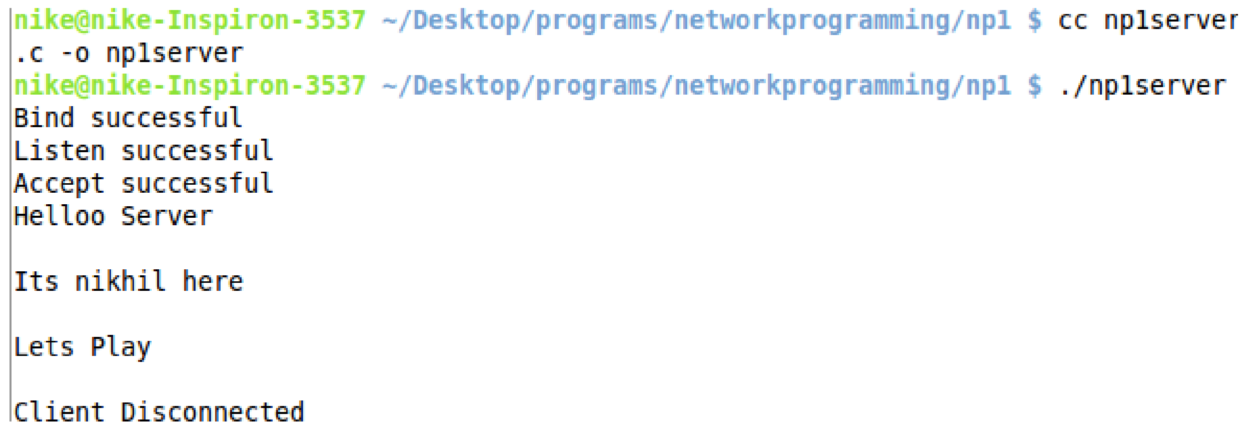
```
                }
        }
        printf("Client Disconnected\n");
        return 0;
}
```

Client Program:

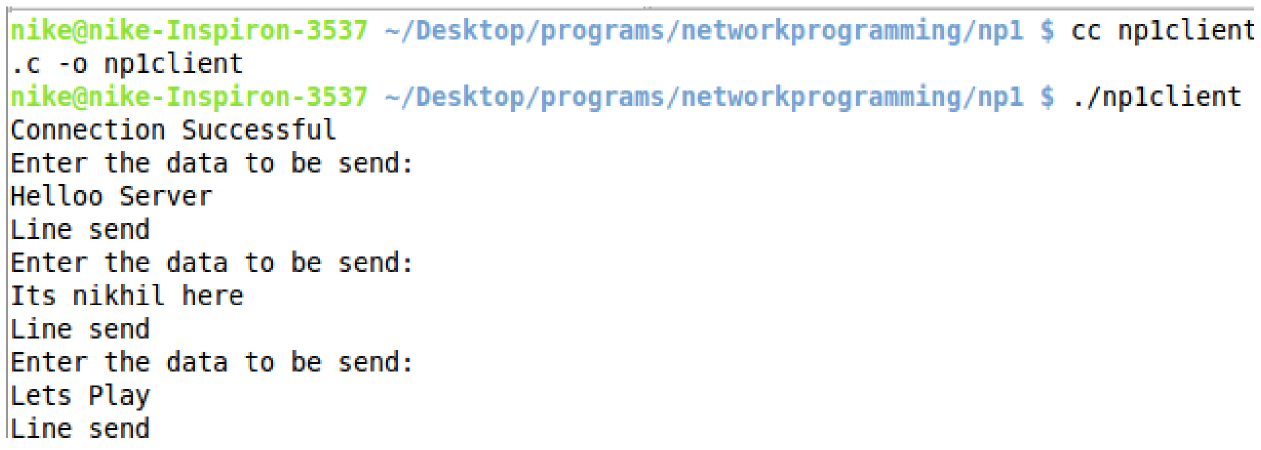
```
#include<sys/types.h>
#include<sys/socket.h>
#include<string.h>
#include<stdio.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define MAXLINE 20
int main()
{
        struct sockaddr_in servaddr;
        char sendline[MAXLINE];
        int n, temp, mysockfd;

        // make socket
        mysockfd = socket(AF_INET, SOCK_STREAM, 0);
        if(mysockfd<0)
        {
                perror("Socket failed");
                return -1;
        }

        // get the server address
        memset(&servaddr, 0, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(5000);

        // connect with the server
        temp = connect(mysockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
        if(temp<0)
        {
                perror("Connection failed");
                return -1;
```

```
	}
	else
	{
		printf("Connection Successful\n");
	}

	// input data to send to server
	printf("Enter the data to be send: \n");
	while(fgets(sendline,MAXLINE,stdin)!=NULL)
	{
		write(mysockfd,sendline,strlen(sendline));
		printf("Line send\n");
		printf("Enter the data to be send: \n");
	}
	exit(0);
	return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np1 $ cc np1server
.c -o np1server
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np1 $ ./np1server
Bind successful
Listen successful
Accept successful
Helloo Server

Its nikhil here

Lets Play

Client Disconnected
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np1 $ cc np1client
.c -o np1client
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np1 $ ./np1client
Connection Successful
Enter the data to be send:
Helloo Server
Line send
Enter the data to be send:
Its nikhil here
Line send
Enter the data to be send:
Lets Play
Line send
```

PROGRAM 2
Reverse String using TCP

Design TCP iterative client and server application to reverse a given input string.

Server Program:

```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<time.h>
#define MAXLEN 200

void reverseString(char input[], char output[]){
	int i=0, len=strlen(input);
	while(len--){
		output[i]=input[len];
		i++;
	}
	output[i]='\0';
}

int main()
{
	struct sockaddr_in servaddr, client;
	char sendData[MAXLEN], rcvData[MAXLEN];
	int n, temp, mysockfd, clientfd;
	int clilen=sizeof(client);

	// create socket at server
	mysockfd = socket(AF_INET, SOCK_STREAM, 0);
	if(mysockfd<0){
		perror("Socket failed");
		return -1;
	}

	// create the server address
```

```
memset(&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(5001);

// bind the server address to the socket
if(bind(mysockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0){
        perror("Bind failed");
        return -1;
}
else{
        printf("Bind successful\n");
}

// listen to the requests with at max 2 requests
if(listen(mysockfd, 2) < 0){
        perror("Listen failed");
        return -1;
}
else{
        printf("Listen successful\n");
}
while(1){
        // accept the clients request
        clientfd = accept(mysockfd, (struct sockaddr *)&client, (unsigned int *)&clilen);
        if(clientfd<0){
                perror("Accept failed");
                return -1;
        }
        else{
                printf("Accept successful\n");
        }

        // get string from client
        n=read(clientfd, rcvData, MAXLEN-1);
        if(n<0){
                perror("Read error in server");
                return -1;
        }
        rcvData[n]='\0';
```

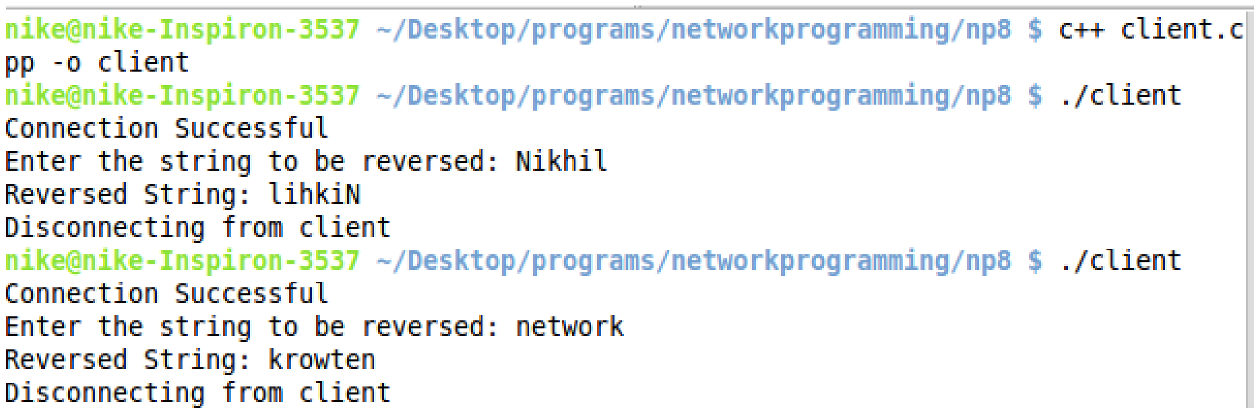
```
		// reverse the string
		reverseString(rcvData, sendData);

		// send reversed string to client
		write(clientfd, sendData, strlen(sendData));
		printf("Reversed String => %s sent to client.\n", sendData);

		printf("Client Disconnected\n");
		close(clientfd);
	}
	return 0;
}
```

Client Program:

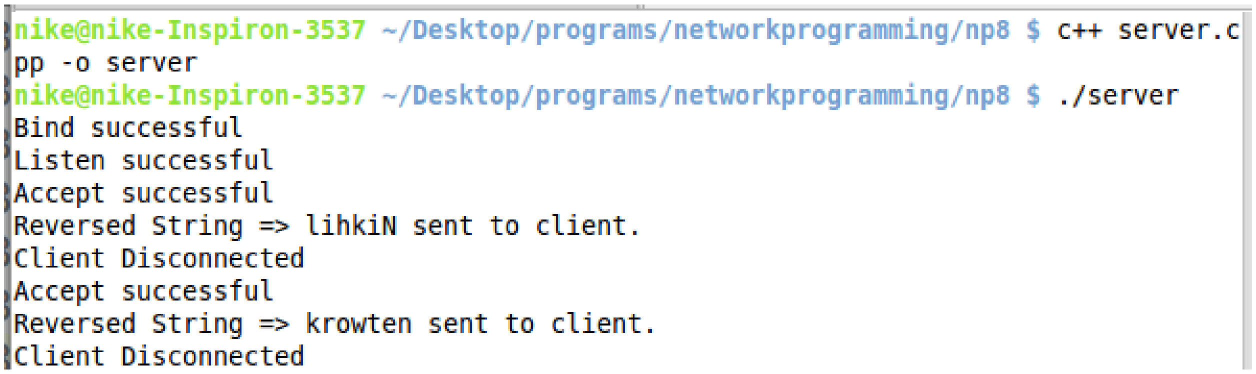
```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define MAXLINE 200

int main(){
	struct sockaddr_in servaddr;
	char rcvline[MAXLINE], sendline[MAXLINE];
	int n, mysockfd;

	// make socket
	mysockfd = socket(AF_INET, SOCK_STREAM, 0);
	if(mysockfd<0){
		perror("Socket failed");
		return -1;
	}

	// get the server address
	memset(&servaddr, 0, sizeof(servaddr));
	servaddr.sin_family = AF_INET;
	servaddr.sin_port = htons(5001);
```

```
        // connect with the server
        if(connect(mysockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0){
                perror("Connection failed");
                return -1;
        }
        else{
                printf("Connection Successful\n");
        }

        // input the string to be reversed
        printf("Enter the string to be reversed: ");
        scanf("%s", sendline);

        // send the string to the server
        write(mysockfd, sendline, strlen(sendline));

        // get the reverse of string from server
        n=read(mysockfd, rcvline, MAXLINE-1);
        if(n<0){
                perror("Read data from server failed");
        }
        else{
                rcvline[n]='\0';
                printf("Reversed String: %s\n", rcvline);
        }
        printf("Disconnecting from client\n");
        return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np8 $ c++ client.c
pp -o client
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np8 $ ./client
Connection Successful
Enter the string to be reversed: Nikhil
Reversed String: lihkiN
Disconnecting from client
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np8 $ ./client
Connection Successful
Enter the string to be reversed: network
Reversed String: krowten
Disconnecting from client
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np8 $ c++ server.c
pp -o server
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np8 $ ./server
Bind successful
Listen successful
Accept successful
Reversed String => lihkiN sent to client.
Client Disconnected
Accept successful
Reversed String => krowten sent to client.
Client Disconnected
```

PROGRAM 3
Date/Time using TCP

Program for date and time server using TCP sockets

Server Program:

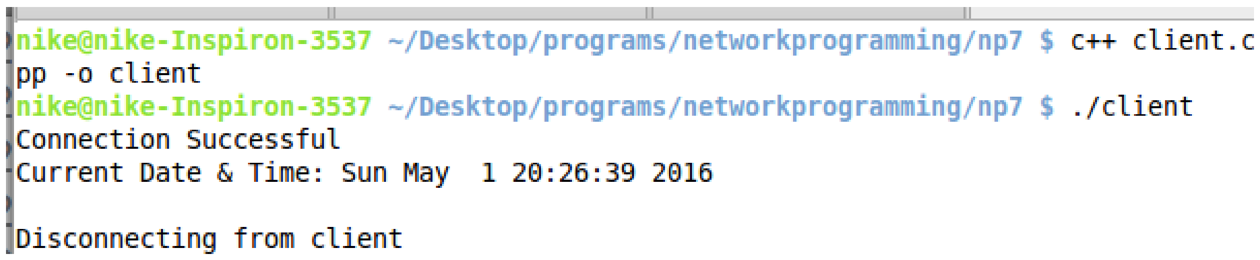
```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#include<time.h>
#define MAXLEN 200

int main()
{
	struct sockaddr_in servaddr, client;
	char sendData[MAXLEN];
	int n, temp, mysockfd, clientfd;
	int clilen=sizeof(client);

	// create socket at server
	mysockfd = socket(AF_INET, SOCK_STREAM, 0);
	if(mysockfd<0){
		perror("Socket failed");
		return -1;
	}

	// create the server address
	memset(&servaddr, 0, sizeof(servaddr));
	servaddr.sin_family = AF_INET;
	servaddr.sin_port = htons(5001);

	// bind the server address to the socket
	if(bind(mysockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0){
		perror("Bind failed");
		return -1;
	}
	else{
		printf("Bind successful\n");
	}

	// listen to the requests with at max 2 requests
	if(listen(mysockfd, 2) < 0){
		perror("Listen failed");
		return -1;
	}
	else{
		printf("Listen successful\n");
	}

	// accept the clients request
	clientfd = accept(mysockfd, (struct sockaddr *)&client, (unsigned int *)&clilen);
	if(clientfd<0){
		perror("Accept failed");
		return -1;
	}
	else{
		printf("Accept successful\n");
	}

	// get current time
	time_t ct;
	ct = time(NULL);
	sprintf(sendData, "%s", ctime(&ct));

	// send time to client
	write(clientfd, sendData, strlen(sendData));
	printf("Current date & time => %s Sent\n",sendData);

	printf("Client Disconnected\n");
	close(clientfd);
	return 0;
}
```

Client Program:

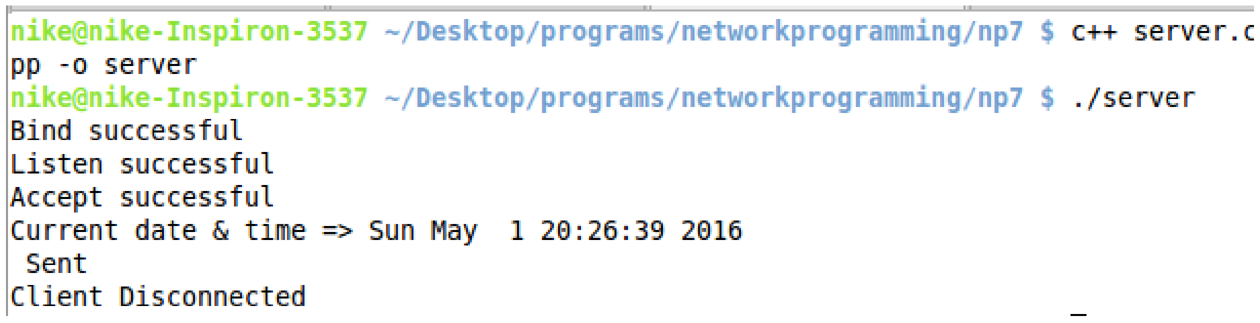
```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define MAXLINE 200

int main(){
	struct sockaddr_in servaddr;
	char rcvline[MAXLINE];
	int n, mysockfd;

	// make socket
	mysockfd = socket(AF_INET, SOCK_STREAM, 0);
	if(mysockfd<0){
		perror("Socket failed");
		return -1;
	}

	// get the server address
	memset(&servaddr, 0, sizeof(servaddr));
	servaddr.sin_family = AF_INET;
	servaddr.sin_port = htons(5001);

	// connect with the server
	if(connect(mysockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0){
		perror("Connection failed");
		return -1;
	}
	else{
		printf("Connection Successful\n");
	}

	// get date/time to server
	printf("Current Date & Time: ");
	while((n=read(mysockfd, rcvline, MAXLINE-1)) > 0){
		rcvline[n]='\0';
```

```
			printf("%s",rcvline);
		}
		printf("\n");
		printf("Disconnecting from client\n");
		return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np7 $ c++ client.c
pp -o client
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np7 $ ./client
Connection Successful
Current Date & Time: Sun May  1 20:26:39 2016

Disconnecting from client
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np7 $ c++ server.c
pp -o server
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np7 $ ./server
Bind successful
Listen successful
Accept successful
Current date & time => Sun May  1 20:26:39 2016
 Sent
Client Disconnected
```

PROGRAM 4
Transfer File Using TCP

Design TCP client and server application to transfer a file.

Server Program:

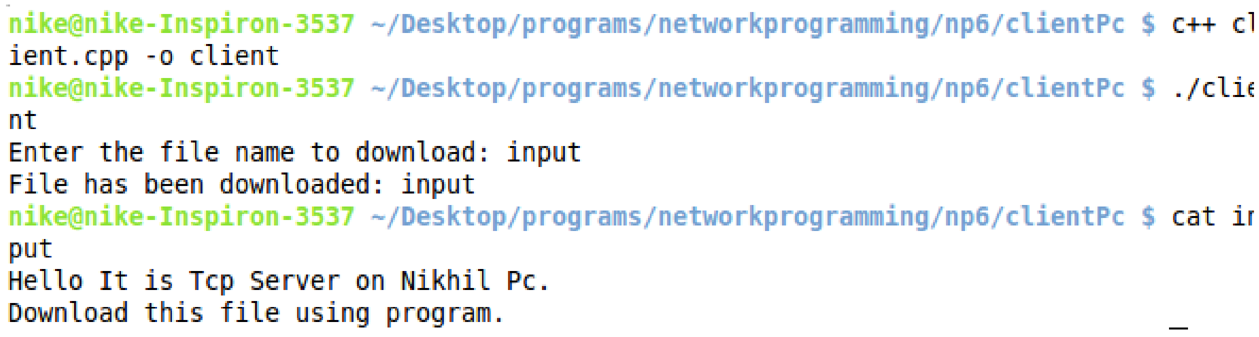
```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<string.h>
#include<stdio.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define MAXLINE 200

int main(){
	int mysockfd, clientSize, clientfd;
	struct sockaddr_in serverAddress, clientAddress;
	char sendData[MAXLINE], fileName[MAXLINE];

	// socket file descriptor at server
	mysockfd=socket(AF_INET, SOCK_STREAM, 0);
	if(mysockfd==-1){
		perror("Socket Socket call failed\n");
		exit(EXIT_FAILURE);
	}

	// create server address
	serverAddress.sin_family=AF_INET;
	serverAddress.sin_port=htons(5000);

	// bind the server address with the server file descriptor
	if(bind(mysockfd, (struct sockaddr *)&serverAddress, (socklen_t)sizeof(serverAddress))
< 0){
		perror("Bind failed");
		return -1;
	}
	// listen to the client requests
	if(listen(mysockfd, 3)<0){
		perror("Listen Failed");
		return -1;
	}

	while(1){
		// accept a client connection
		if((clientfd=accept(mysockfd, (struct sockaddr *)&clientAddress, (socklen_t
*)&clientSize)) < 0){
			perror("Accept Failed");
			return -1;
		}

		printf("New Client Connected\n");

		// receive the name of file
		if(recv(clientfd, fileName, MAXLINE, 0) < 0){
			perror("Receive failed");
		}
		printf("File requested by client: %s\n", fileName);

		// open the file and send it to client
		FILE *input = fopen(fileName, "r");
		if(input==NULL){
			printf("Requested file not available on server\n");
			strcpy(sendData, "File not available\n");
			send(clientfd, sendData, strlen(sendData)+1, 0);
			return -1;
		}
		else{
			while(fgets(sendData, MAXLINE, input)){
				write(clientfd, sendData, strlen(sendData));
			}
			printf("Requested File %s sent to client.\n",fileName);
		}

		// close the files and connection
		fclose(input);
```

```
                close(clientfd);
                printf("Client Disconnected\n\n");
        }
        return 0;
}
```

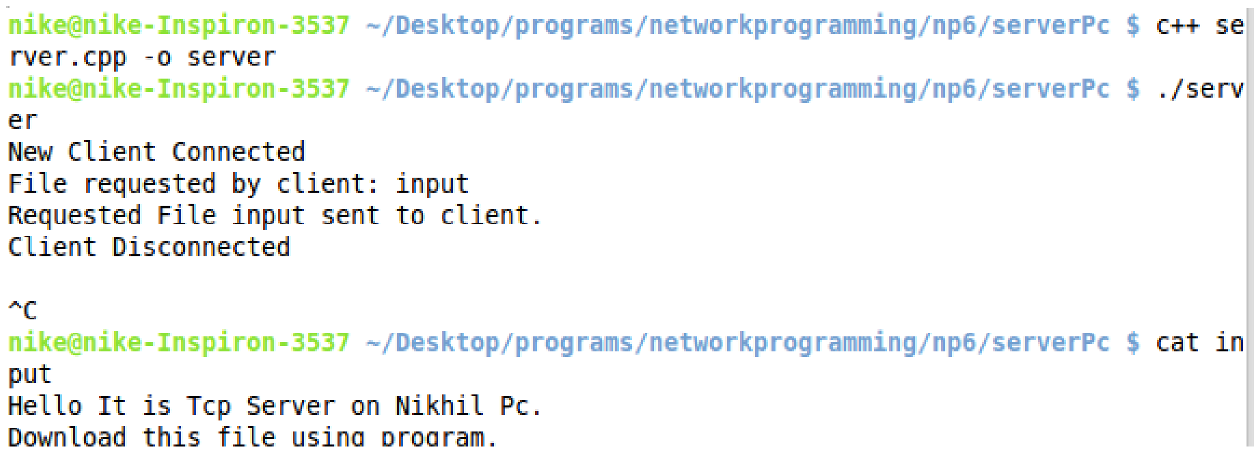
Client Program:

```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<string.h>
#include<stdio.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define MAXLINE 200

int main(){
        int mysockfd, len;
        unsigned int serverSize;
        char sendData[MAXLINE],recvData[MAXLINE],fileName[MAXLINE];
        struct sockaddr_in serverAddress;

        // socket at client
        mysockfd=socket(AF_INET, SOCK_STREAM, 0);
        if(mysockfd==-1){
                perror("Socket client failed");
                exit(-1);
        }

        // server address
        memset(&serverAddress, 0, sizeof(serverAddress));
        serverAddress.sin_family = AF_INET;
        serverAddress.sin_port = htons(5000);

        // connect to the server
        if(connect(mysockfd, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) < 0){
                perror("Connect failed");
                return -1;
        }

        // get the filename to be downloaded
        printf("Enter the file name to download: ");
        scanf("%s",fileName);

        // send the file name to the server
        if(send(mysockfd, fileName, strlen(fileName), 0) < 0){
                perror("Sending failed");
                return -1;
        }

        // download the file
        FILE *download = fopen(fileName, "w");
        if(download==NULL){
                printf("File opening failed\n");
                return -1;
        }
        else{
                while((len=read(mysockfd, recvData, MAXLINE-1)) > 0){
                        recvData[len]='\0';
                        fputs(recvData, download);
                }
        }
        fclose(download);

        printf("File has been downloaded: %s\n",fileName);
        return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np6/clientPc $ c++ cl
ient.cpp -o client
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np6/clientPc $ ./clie
nt
Enter the file name to download: input
File has been downloaded: input
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np6/clientPc $ cat in
put
Hello It is Tcp Server on Nikhil Pc.
Download this file using program.
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np6/serverPc $ c++ se
rver.cpp -o server
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np6/serverPc $ ./serv
er
New Client Connected
File requested by client: input
Requested File input sent to client.
Client Disconnected

^C
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np6/serverPc $ cat in
put
Hello It is Tcp Server on Nikhil Pc.
Download this file using program.
```

PROGRAM 5

Transfer File using UDP

UDP client and server application to transfer a file

Server Program:

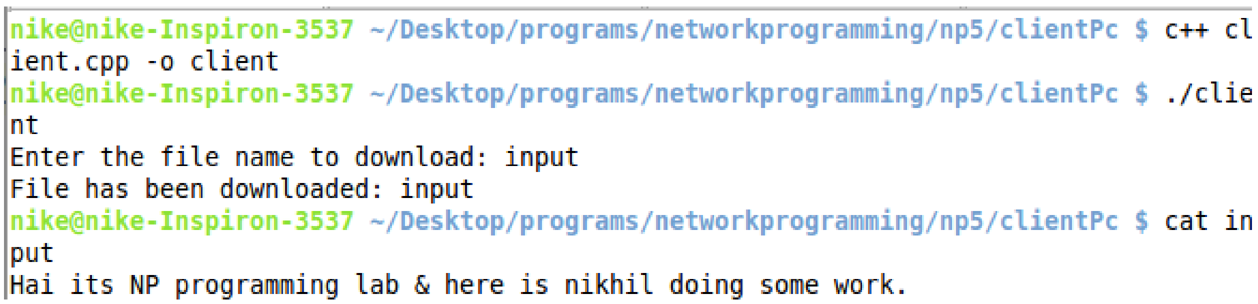
```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<string.h>
#include<stdio.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define MAXLINE 200

int main(){
	int mysockfd, clientSize;
	struct sockaddr_in serverAddress, clientAddress;
	char sendData[MAXLINE], fileName[MAXLINE];

	// socket file descriptor at server
	mysockfd=socket(AF_INET, SOCK_DGRAM, 0);
	if(mysockfd==-1){
		perror("Socket Socket call failed\n");
		exit(EXIT_FAILURE);
	}

	// create server address
	serverAddress.sin_family=AF_INET;
	serverAddress.sin_port=htons(5000);

	// bind the server address with the server file descriptor
	if(bind(mysockfd, (struct sockaddr *)&serverAddress, (socklen_t)sizeof(serverAddress))
< 0){
		perror("Bind failed");
		return -1;
	}

	// receive the name of file
	if(recvfrom(mysockfd, fileName, MAXLINE, 0, (struct sockaddr *)&clientAddress,
(socklen_t *)&clientSize) < 0){
		perror("Receive failed");
	}
	printf("File requested by client: %s\n", fileName);

	// open the file and send it to client
	FILE *input = fopen(fileName, "r");
	if(input==NULL){
		printf("Requested file not available on server\n");
		strcpy(sendData, "File not available\n");
		sendto(mysockfd, sendData, strlen(sendData)+1, 0, (struct sockaddr
*)&clientAddress, (socklen_t)clientSize);
		return -1;
	}
	else{
		while(fgets(sendData, MAXLINE, input)){
			sendto(mysockfd, sendData, MAXLINE, 0, (struct sockaddr
*)&clientAddress, (socklen_t)clientSize);
		}
		printf("Requested File %s sent to client.\n",fileName);
	}
	fclose(input);
	return 0;
}
```

Client Program:

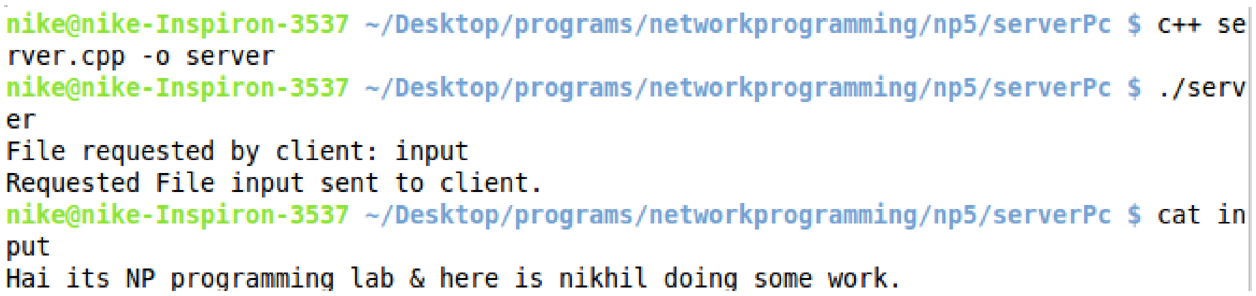
```
#include<bits/stdc++.h>
using namespace std;
#include<sys/types.h>
#include<sys/socket.h>
#include<string.h>
#include<stdio.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#define MAXLINE 200

int main(){
	int mysockfd, len;
	unsigned int serverSize;
	char sendData[MAXLINE],recvData[MAXLINE],fileName[MAXLINE];
	struct sockaddr_in serverAddress;

	// socket at client
	mysockfd=socket(AF_INET, SOCK_DGRAM, 0);
	if(mysockfd==-1){
		perror("Socket client failed");
		exit(-1);
	}

	// server address
	memset(&serverAddress, 0, sizeof(serverAddress));
	serverAddress.sin_family = AF_INET;
	serverAddress.sin_port = htons(5000);

	// get the filename to be downloaded
	printf("Enter the file name to download: ");
	scanf("%s",fileName);

	// send the file name to the server
	serverSize=sizeof(serverAddress);
	if(sendto(mysockfd, fileName, strlen(fileName), 0, (struct sockaddr*)&serverAddress,
(socklen_t)serverSize) < 0){
		perror("Sending failed");
		return -1;
	}

	// download the file
	FILE *download = fopen(fileName, "w");
	if(download==NULL){
		printf("File opening failed\n");
		return -1;
	}
	else{
		while(recvfrom(mysockfd, recvData, MAXLINE, 0, (struct sockaddr
*)&serverAddress, (socklen_t *)&serverSize) > 0){
			fputs(recvData, download);
// here break should not be done but in udp it recvfrom cannot stop as it cannot find EOF
// break is done assuming file is less than MAXLINE size
```

```
                    break;
                }
        }
        fclose(download);

        printf("File has been downloaded: %s\n",fileName);
        return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np5/clientPc $ c++ cl
ient.cpp -o client
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np5/clientPc $ ./clie
nt
Enter the file name to download: input
File has been downloaded: input
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np5/clientPc $ cat in
put
Hai its NP programming lab & here is nikhil doing some work.
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np5/serverPc $ c++ se
rver.cpp -o server
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np5/serverPc $ ./serv
er
File requested by client: input
Requested File input sent to client.
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/np5/serverPc $ cat in
put
Hai its NP programming lab & here is nikhil doing some work.
```

PROGRAM 6
PIPE

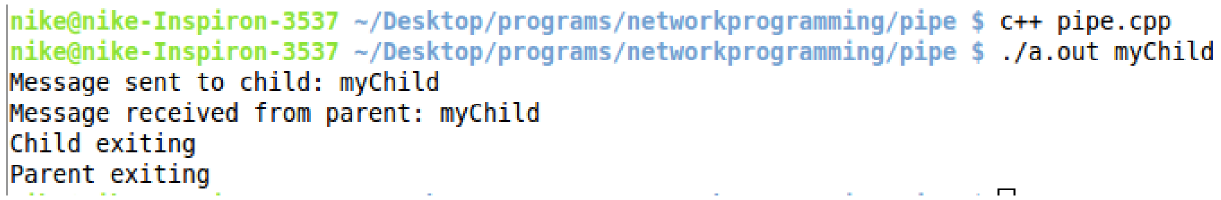
Creation of a one way pipe in a single process

Program:

```
#include<sys/wait.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(int argc, char *argv[]){
	// 1 argument i.e. message to be sent to child
	if(argc!=2){
		printf("1 argument is required.\n");
		exit(EXIT_FAILURE);
	}

	int pipefd[2];
	pid_t cpid;
	char buf;

	if(pipe(pipefd)!=-1){
		// pipe successful

		cpid=fork();
		if(cpid==-1){
			// fork failed
			printf("fork failed\n");
			exit(EXIT_FAILURE);
		}

		if(cpid==0){
			// child process
			// close the write descriptor in the child process
			close(pipefd[1]);
			printf("Message received from parent: ");
			while(read(pipefd[0], &buf, 1)){
				printf("%c",buf);
```

```
                        }
                        printf("\n");
                        printf("Child exiting\n");
                        return 0;
                }
                else{
                        // parent process
                        // close the read descriptor in the parent process
                        close(pipefd[0]);
                        printf("Message sent to child: %s\n", argv[1]);
                        write(pipefd[1], argv[1], strlen(argv[1]));
                        close(pipefd[1]);
                        wait(NULL);
                        printf("Parent exiting\n");
                        return 0;
                }
        }
        else{
                // pipe failure
                printf("Pipe failed\n");
                exit(EXIT_FAILURE);
        }
        return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/pipe $ c++ pipe.cpp
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/pipe $ ./a.out myChild
Message sent to child: myChild
Message received from parent: myChild
Child exiting
Parent exiting
```

PROGRAM 7
FIFO

To make a Server client for receiving and sending messages using FIFO

Creating FIFO:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
        if(argc!=2){
                printf("Enter the name of the fifo to be made as an argument\n");
                exit(EXIT_FAILURE);
        }

        // make fifo
        if(mkfifo(argv[1], 0777)==-1){
                printf("unable to make fifo\n");
                exit(EXIT_FAILURE);
        }

        printf("Fifo %s made.\n", argv[1]);
        return 0;
}
```

FIFO Write Program:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
        if(argc!=2){
                printf("Enter the name of the fifo to write data\n");
                exit(EXIT_FAILURE);
```

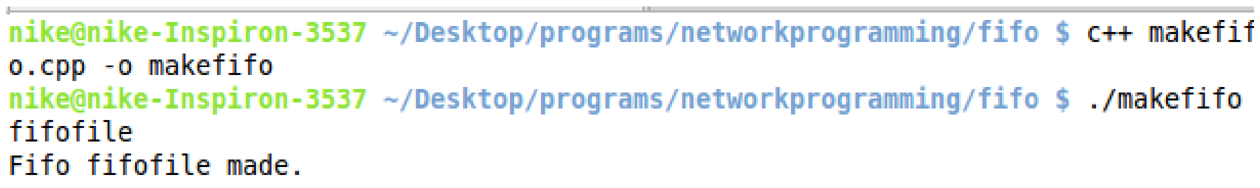
```
	}

	char buffer[100];
	// open the fifo file to write data
	FILE *myfile=fopen(argv[1], "w");
	if(myfile==NULL){
		printf("unable to open fifo file\n");
		exit(EXIT_FAILURE);
	}

	printf("Enter the data to be sent: \n");
	fgets(buffer, 100, stdin);
	fputs(buffer, myfile);
	printf("Data sent.\n");

	fclose(myfile);
	return 0;
}
```

FIFO Read Program:

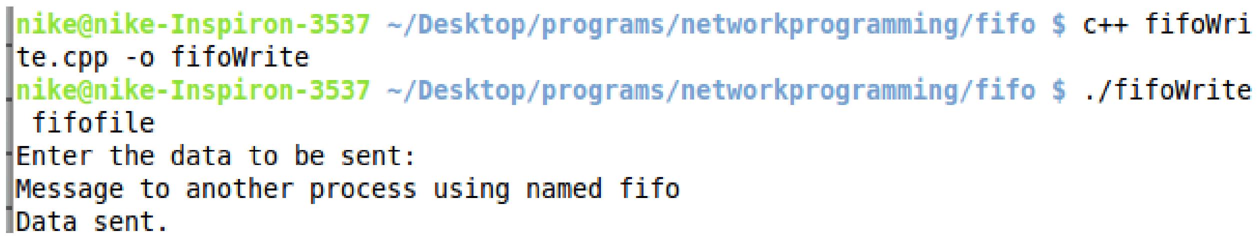
```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
	if(argc!=2){
		printf("Enter the name of the fifo to read data\n");
		exit(EXIT_FAILURE);
	}

	char buffer[100];
	// open the fifo file to read data
	FILE *myfile=fopen(argv[1], "r");
	if(myfile==NULL){
		printf("unable to open fifo file\n");
		exit(EXIT_FAILURE);
	}
```

```
        fgets(buffer, 100, myfile);
        printf("Data received: %s\n",buffer);

        fclose(myfile);
        return 0;
}
```

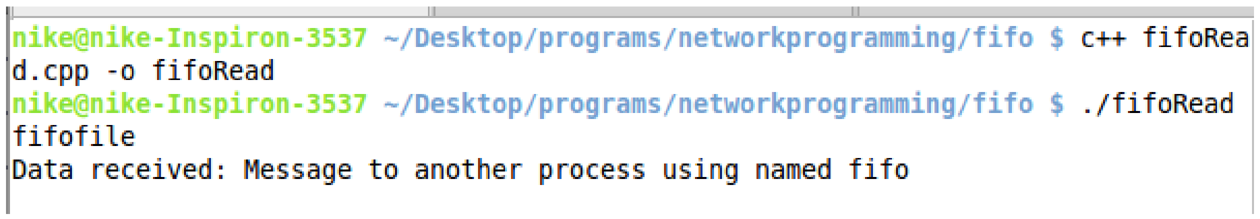
Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/fifo $ c++ makefif
o.cpp -o makefifo
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/fifo $ ./makefifo
fifofile
Fifo fifofile made.
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/fifo $ c++ fifoWri
te.cpp -o fifoWrite
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/fifo $ ./fifoWrite
 fifofile
Enter the data to be sent:
Message to another process using named fifo
Data sent.
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/fifo $ c++ fifoRea
d.cpp -o fifoRead
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/fifo $ ./fifoRead
fifofile
Data received: Message to another process using named fifo
```

PROGRAM 8
Message Queue

Program to implement Message Queue where a message is sent from 1 process to another. Also display information about the message queue.

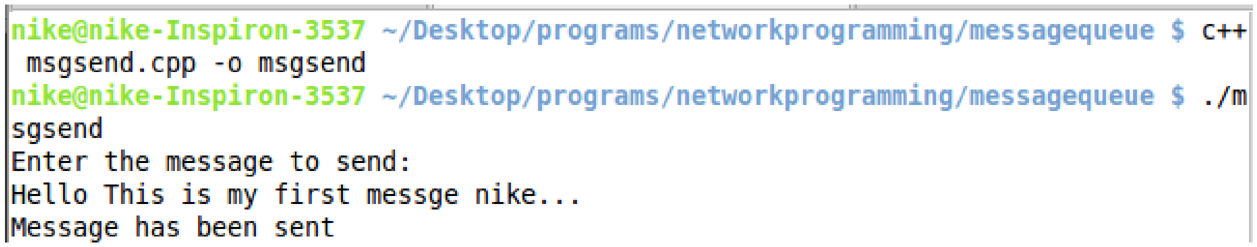
Message Send Program:

```
#include<bits/stdc++.h>
using namespace std;
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MSG_KEY 34
#define MAX_LEN 100

typedef struct mymsgbuf{
   long mtype;
   char mtext[MAX_LEN];
}msg;

int main(int argc, char const *argv[])
{
        int msqQue, temp;
        msg sendMsg;

        // create the msg queue
        msqQue = msgget(MSG_KEY, IPC_CREAT | 0666);
        if(msqQue==-1){
                perror("Msq Queue failed");
                return -1;
        }

        // get the msg from user
        printf("Enter the message to send:\n");
        scanf("%[^\n]",sendMsg.mtext);

        // send the message
        if(msgsnd(msqQue, &sendMsg, strlen(sendMsg.mtext)+1, 0) < 0){
                perror("Message Sending failed");
                return -1;
        }
        printf("Message has been sent\n");
        return 0;
}
```

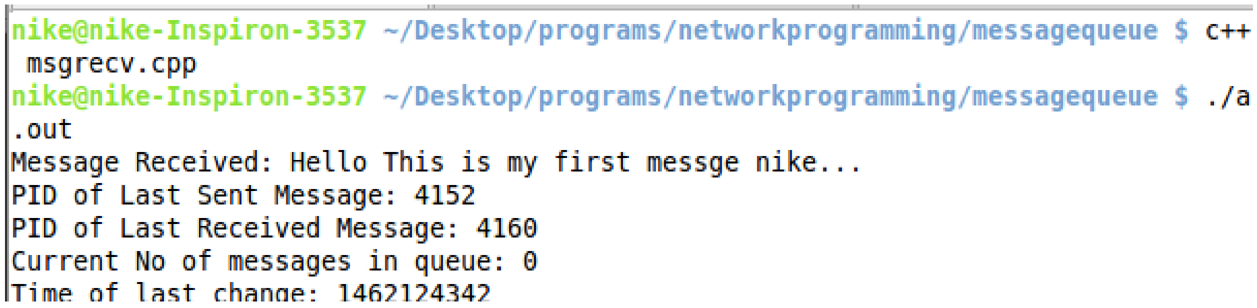
Message Receive Program:

```
#include<bits/stdc++.h>
using namespace std;
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MSG_KEY 34
#define MAX_LEN 100

typedef struct mymsgbuf{
   long mtype;
   char mtext[MAX_LEN];
}msg;

int main(int argc, char const *argv[])
{
        int msgQue, temp;
        msg rcvMsg;
        struct msqid_ds myqueue;

        // create the msg queue
        msgQue = msgget(MSG_KEY, IPC_CREAT | 0666);
        if(msgQue==-1){
                perror("Msq Queue failed");
                return -1;
        }

        // receive the message
        if(msgrcv(msgQue, &rcvMsg, MAX_LEN, 0, 0) < 0){
                perror("Message Receiving failed");
                return -1;
        }
        // print the message
        printf("Message Received: %s\n", rcvMsg.mtext);

        // print data about queue
        msgctl(msgQue, IPC_STAT, &myqueue);
        printf("PID of Last Sent Message: %d\n", myqueue.msg_lspid);
        printf("PID of Last Received Message: %d\n", myqueue.msg_lrpid);
        printf("Current No of messages in queue: %lu\n", myqueue.msg_qnum);
        printf("Time of last change: %ld\n", myqueue.msg_ctime);
        return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/messagequeue $ c++
 msgsend.cpp -o msgsend
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/messagequeue $ ./m
sgsend
Enter the message to send:
Hello This is my first messge nike...
Message has been sent
```



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/messagequeue $ c++
 msgrecv.cpp
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/messagequeue $ ./a
.out
Message Received: Hello This is my first messge nike...
PID of Last Sent Message: 4152
PID of Last Received Message: 4160
Current No of messages in queue: 0
Time of last change: 1462124342
```

PROGRAM 9
Semaphore

To perform Semaphore Operations

Semaphore Process1 Program:

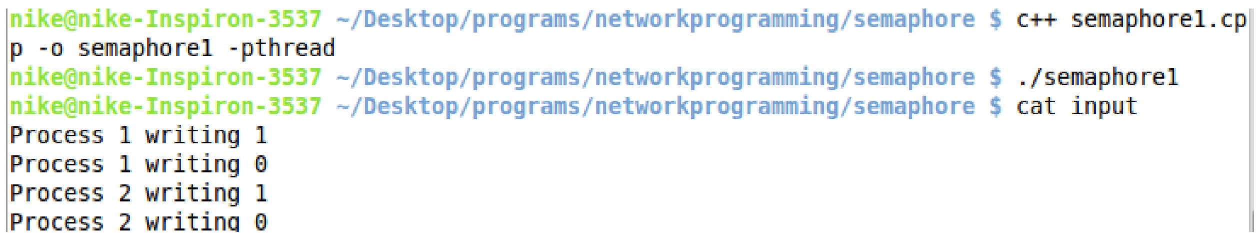
```
#include<bits/stdc++.h>
using namespace std;
#include<fcntl.h>
#include<sys/stat.h>
#include<semaphore.h>
#include<unistd.h>

int main(){
        sem_t *semAddress;
        int i;

        // A semaphore is created with name as "hai" and initial value of resources as 1
        semAddress=sem_open("hai", O_CREAT, 0777, 1);
        if(semAddress==NULL){
                perror("Error while opening semaphore");
                return -1;
        }
        sleep(1);
        // write to the file when semaphore is free
        for(i=0;i<2;i++){
                sem_wait(semAddress);
                FILE *input = fopen("input", "a");
                fprintf(input, "Process 1 writing %d\n",i);
                sem_post(semAddress);
        }

        return 0;
}
```

Semaphore Process2 Program:

```
#include<bits/stdc++.h>
using namespace std;
#include<fcntl.h>
```

```
#include<sys/stat.h>
#include<semaphore.h>

int main(){
	sem_t *semAddress;
	int i;

	semAddress=sem_open("hai", O_EXCL);
	if(semAddress==NULL){
		printf("Error while opening semaphore %d\n",errno);
		return -1;
	}

	// write to file when semaphore is free
	for(i=0;i<2;i++){
		sem_wait(semAddress);
		FILE *input = fopen("input", "a");
		fprintf(input, "Process 2 writing %d\n",i);
		sem_post(semAddress);
	}

	return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/semaphore $ c++ semaphore1.cp
p -o semaphore1 -pthread
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/semaphore $ ./semaphore1
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/semaphore $ cat input
Process 1 writing 1
Process 1 writing 0
Process 2 writing 1
Process 2 writing 0
```

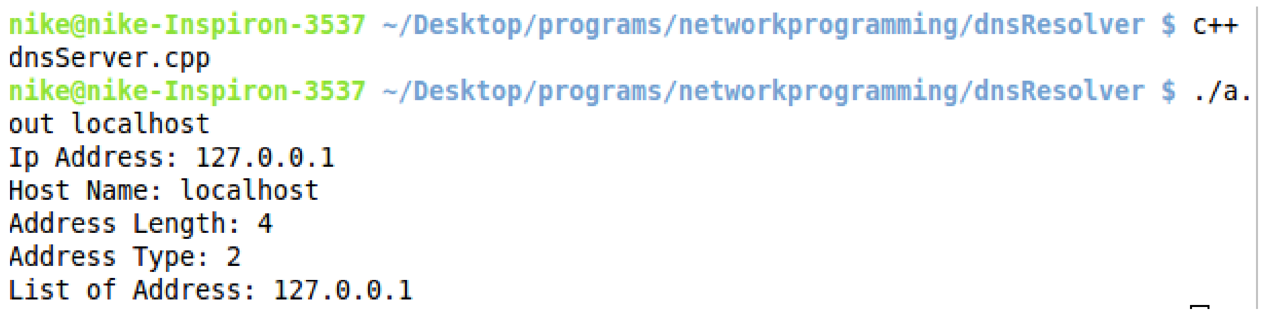
PROGRAM 10
DNS Server

DNS Server to resolve a given host name

Program:

```
#include<stdio.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<netinet/in.h>
int main(int argc, char* argv[])
{
	struct hostent* host;
	struct in_addr h_addr;

	if(argc!=2){
		printf("Requires 1 argument\n");
	}

	// get the host
	if((host=gethostbyname(argv[1]))==NULL){
		printf("Nslookup Failed for %s\n",argv[1]);
	}

	// convert ip address from network byte order to dotted decimal notation
	h_addr.s_addr = *((unsigned long*)host->h_addr_list[0]);
	printf("Ip Address: %s\n", inet_ntoa(h_addr));

	// print the details
	printf("Host Name: %s\n",host->h_name);
	printf("Address Length: %d\n",host->h_length);
	printf("Address Type: %d\n",host->h_addrtype);
	printf("List of Address: %s\n",inet_ntoa(h_addr_list[0]));
	return 0;
}
```

Output:



```
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/dnsResolver $ c++
dnsServer.cpp
nike@nike-Inspiron-3537 ~/Desktop/programs/networkprogramming/dnsResolver $ ./a.
out localhost
Ip Address: 127.0.0.1
Host Name: localhost
Address Length: 4
Address Type: 2
List of Address: 127.0.0.1
```