Date

Expt. No. 6. Page No. 13

6. Implementation of concurrent & iterative echo server using connectionless and connection oriented socket system calls.

* A new header file is created which will be included in all the subprograms.

```
// Headers.h

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

#define BACKLOG 5
#define BUFSIZE 1024.
```

Teacher's Signature :

Output:

```
$ ./ concurrent serv
 Listen called
   Accepted
   Accepted
    Sent
    Sent
```

Server

```
$ ./ connclient  127.0.0.1
    Socket created : 3
    Enter message :  Typo occurred
     Typo occurred
  $.
```

Client 1

```
$ ./ connclient  127.0.0.1
     Socket Created : 3
      Enter message ;  Hi
       Hi
$
```

Client 2.

a) Concurrent TCP server - client

```
//server
#include "Headers.h";
#define PORT 5000
#define IP INADDR_ANY

void servfunc (int sockfd)
{
        char buf[BUFSIZE];
        int cnt;
        bzero (buf, BUFSIZE);
        cnt = recv (sockfd, buf, BUFSIZE, 0);
        if (cnt<0)
        {       exit(0); }
        else if (cnt>0)
        {
                send (sockfd, buf, cnt, 0);
                bzero (buf, BUFSIZE);
        }
        printf ("Sent\n");

}

int main ()
{
        struct sockaddr_in serv;
        int listenfd, acceptfd, fval;
        bzero (&serv, sizeof(serv));
```

//cont...

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6a

Title _____

```
// Server
    serv.sin_family = AF_INET;
    serv.sin_port = htons(PORT);
    serv.sin_addr.s_addr = htons(IP);
    listenfd=Socket (AF_INET, SOCK_STREAM, 0);
    if (bind (listenfd, (struct sockaddr *) &serv,
                    sizeof (serv)) == -1 ) {
        printf( "Error bind "); exit(0);
    }
    listen (listenfd, BACKLOG, 0 );
    printf ( "Listen called");
    while (1)
    {
        acceptfd = accept (listenfd, NULL, NULL);
        if (acceptfd == -1)
        { printf ("Error accepting");
          exit(0);
        } printf ( "Accepted");
```

Signature of
Teacher incharge

```
        fval = fork()
    6   if (fval == 0)
        {
            close(listenfd);
            servfunc(acceptfd);
            close(acceptfd);
            exit(0);
        }
        close(acceptfd);
    }
    return 0;
} //end of server.
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6a

Title Client

```
// Client
#include "Headers.h"
# define IP "127.0.0.1"
# define PORT 5000

int main()
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
        bzero(&serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_port = htons(PORT);
        if(inet_pton(AF_INET, IP, (void *)&serv.sin_
                                        addr) == 0)
        {   printf("Invalid addr \n");
            exit(0);
        }
        printf("socket created");
```

Signature of
Teacher incharge

```
if (connect(sockfd, (struct sockaddr *) &serv,
                    sizeof(serv)) == -1 )
  {
      printf(" Error connecting");
       exit(0);
  }
printf ("Enter Message:\n");
bzero(buf, BUFSIZE);
fgets( buf, BUFSIZE, stdin);
  send (sockfd, buf, strlen(buf),0);
    bzero(buf);
  cnt = recv( sockfd, buf, BUFSIZE, 0);
    if (cnt>0)
     { write(1, buf, cnt); }
       close (sockfd);
    return 0;
} // end of 6a.
```

Output:

```
$ ./ iterativeconnserv
  Listen called
   Accepted
   Accepted.
```

Server

```
$ ./ connclient 127.0.0.1
     Socket created :3
      Enter message: Good afternoon
      Good Afternoon
$
```

Client 1

```
$ ./ connclient 127.0.0.1
     Socket created : 3
      Enter message : Hello
       Hello.
```

Client 2

6.b) Iterative TCP server (echo server) - client.

```
// Server - connection oriented, iterative
#include "Headers.h"

#define PORT 5000
#define IP INADDR_ANY

void servfunc (int sockfd)
{
        char buf[BUFSIZE]
        int cnt;
        bzero( buf, BUFSIZE);
        cnt = recv( sockfd, buf, BUFSIZE, 0);
        if( cnt>0)
        {
                send (sockfd, buf, cnt, 0);
                bzero( buf, BUFSIZE);
        }
}

int main ()
{
        struct sockaddr_in srv;
        int listenfd, acceptfd, fval;
        bzero (&srv, sizeof(srv));
        srv.sin_family = AF_INET;
        srv.sin_port = htons( PORT);
        srv.sin_addr.s_addr = htonl (IP);
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6B

Title _____

```
//server

listenfd = ~~listen~~ socket (AF_INET, SOCK_STREAM, 0);
if ( bind (listenfd, (struct sockaddr*) &srv,
        sizeof (srv)) == -1)
{
    printf ( "Bind error");
    exit (0);
}
if (listen ( listenfd, BACKLOG) < 0)
{
    printf ( "Listen error");
    exit (0);
}
printf ( "Listen called");

// cont...
```

Signature of
Teacher incharge

```
// servfunc to be called.

while (1)
{
        acceptfd = accept (listenfd, NULL, NULL)
        if (acceptfd == -1)
        {
            printf ("Error accepting");
            exit(0);
        }
        printf ("Accepted");
        servfunc (acceptfd);
        close (acceptfd);
    }
    return 0;
}
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 8B

Title Client

```
// Client code - iterative, connection oriented.
#include "Headers.h"
#define PORT 5000
#define IP "127.0.0.1"

int main (int argc, char * argv[])
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;

        if (argc != 2)
        {
            printf ("Format : ./a.out <IP>\n");
            exit(0);
        }

        bzero( &serv, sizeof (serv));
        serv. sin_family=AF_INET;
        serv. sin_port = htons
                        (PORT);
```

Signature of
Teacher incharge

```
if ( inet_pton (AF_INET, argv[1], (void *) &servaddr.sin_
                addr ) == 0 )
{
        printf ("Invalid address");
        exit(0);
}
sockfd = socket (AF_INET, SOCK_STREAM, 0);
if (connect ( sockfd, (struct sockaddr *) &serv,
{           sizeof (serv) ) == -1).
        printf ("unable to connect");
        exit(0);
}
printf ("Enter message: \n");
bzero (buf, BUFSIZE);
fgets( buf, BUFSIZE, stdin );
send ( sockfd, buf, strlen(buf), 0);
bzero ( buf, BUFSIZE);
cnt = recv ( sockfd, buf, BUFSIZE, 0);
// cont..
```

```
// client cont...

if (cnt < 0)
  {
      printf ("Error");
  }
else if (cnt > 0)
  {
      write (1, buf, cnt);
  }
  close (sockfd);

  return 0;
}
// end of client for 6b.
```

Output :

Server

```
$ ./iterativeconnlessserv
Accepted
Sent
Accepted
Sent
```

Client 1

```
$ ./connlessclient 127.0.0.1
socket created : 3
Enter message : Helloworld
Helloworld.
$
```

Client 2.

```
$./connlessclient 127.0.0.1
socket created : 3
Enter message : No typo
No typo
```

6.c) UDP echo

```
// server
#include "Headers.h"
# define PORT 5000
# define IP INADDR_ANY

void servfunc ( int sockfd, struct sockaddr_in *client , int *len)
{
        char buf[BUFSIZE] ;
        int cnt;
        bzero ( buf, BUFSIZE);
        cnt = recvfrom ( sockfd, buf, BUFSIZE, 0 ,
            (struct sockaddr *) client , (socklen_t*)len );
        if ( cnt >0 )
                sendto ( sockfd, buf, cnt, 0 ,
                    (struct sockaddr *) client, *len );
        else
          {  printf (" Nothing received ");
             exit(0);
          }
         printf ( " sent \n" );
}

int main ()
{
        struct sockaddr_in serv;
        int sockfd, fval ;
        bzero( &serv, sizeof(serv) );
```

Teacher's Signature :

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6

Title 6.c)

```
// cont... server
serv.sin_family = AF_INET;
serv.sin_port = htons(PORT);
serv.sin_addr.s_addr = htonl(IP);
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (bind(sockfd, (struct sockaddr*) &serv,
                sizeof(serv)) == -1)
{
    printf("Bind failed");
    exit(0);
}
struct sockaddr_in client;
int len;
while(1)
{
    len = sizeof(client);
    bzero(&client, len);
    servfunc(sockfd, &client, &len);
}
return 0;
} // end of server.
```

Signature of
Teacher incharge

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M.C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 6

Title 6. C

```
// Client
#include "Headers.h"
#define PORT 5000
#define IP "127.0.0.1"

int main (int argc, char * argv[])
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
        if (argc != 2)
        {
            printf ("Format: ./a.out <IP> \n");
            exit(0);
        }
        bzero( & serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_addr
        serv.sin_port = htons( PORT);
//cont...
```

Signature of
Teacher incharge

Expt. No. ----------

```
if ( inet_pton( AF_INET, IP, (void *) & serv.sin_addr
                == 0 )
{
        printf( "Invalid address \n");
        exit(0);
}
sockfd = socket( AF_INE, SOCK_DGRAM, 0);
printf (" Socket created ");
printf (" Enter the message\n");
bzero ( buf, BUFSIZE );
fgets( buf, BUFSIZE, stdin);
cnt = sendto( sockfd, buf, strlen(buf), 0,
      (struct sockaddr *) & serv, sizeof (serv))
if (cnt<0)
{
        printf ("Unable to send \n");
        exit(0);
}
bzero ( buf, BUFSIZE);
cnt = recvfrom ( sockfd, buf, BUFSIZE, 0,
                 NULL, NULL) ;
if (cnt>0)
  write ( 1, buf, cnt);
        close (sockfd);
        return 0;
}
//end.
```