Program No.14 Concurrent Time Server

Aim:-

To Implement a Concurrent time server using UDP as transport layer protocol by executing the program at remote server. Client sends a time request to Server and Server sends its system time back to the client. Client displays the result.

Problem description – Using UDP socket, create connection between multiple clients and single time server.

Algorithm – UDP SERVER

1. Create a socket for UDP using the function call, socket(AF_INET, SOCK_DGRAM, 0);
2. Declare a time object variable ct of data type, time_t
3. The bzero() function places null bytes of memory area pointed to by local.
   bzero((char*)&servaddr,sizeof(servaddr));
4. Initialize the structure sockaddr_in members of sin_family, sin_addr, sin_port
5. Bind the socket to its port using bind(s,(struct sockaddr*)&servaddr,sizeof(servaddr)
6. Receive time request from client using recvfrom(s,buffer,1024,0,(struct sockaddr*)&cliaddr,&t
7. Initializes ct=time(NULL) and Prints the current date and time by calling ctime(&ct).
8. Child process is created. Parent process stops listening for new connections. Child will continue to accept TIME requests from other clients, since it is a concurrent server. The main (parent) process now handles the connected client.
9. After clearing the buffer memory area using memset() function, TIME request is received from client using recvfrom(s,buffer,1024,0,(struct sockaddr*)&cliaddr,&t
10. Prints the formatted string TIME to buffer.
11. Sends back UPDATED CURRENT TIME to client using sendto(s,buffer,sizeof(buffer),0,(struct sockaddr*)&cliaddr,sizeof(cliaddr)
12. Close the socket using close(int sockfd) function.

Algorithm – UDP CLIENT

1. Create a socket for UDP using the function call, socket(AF_INET, SOCK_DGRAM, 0);
2. The bzero() function places null bytes of memory area pointed to by local.
   bzero((char *)&local,sizeof(local));
3. Initialize the structure sockaddr_in members of sin_family, sin_addr, sin_port
4. Bind the socket to its port using bind(s,(struct sockaddr *)&local,sizeof(local))
5. The bzero() function places null bytes of memory area pointed to by servaddr.
   bzero((char *)&servaddr,sizeof(local));
6. Client sends TIME request to server using sendto(s,buffer,sizeof(buffer),0,(struct sockaddr*)&servaddr,sizeof(servaddr)
7. Client receives TIME response from server using using recvfrom() function as follows:
   recvfrom(s,buffer,1024,0,(struct sockaddr *)&servaddr,&t)
8. Prints the received message in client's terminal.

CONCURRENT CLIENT PROGRAM – conclient.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<string.h>
#include<netinet/in.h>
#include<errno.h>
#include<time.h>
int main(int argc,char *argv[]) {
int n,s,t;
struct sockaddr_in servaddr,local,rem;
char buffer[1024];
if(argc<3)
{
printf("usage:client<server-addr><port>");
exit(0);
}
if((s=socket(AF_INET,SOCK_DGRAM,0))<0)
{
perror("error in socket creation");
exit(0);
}
bzero((char *)&local,sizeof(local));
local.sin_family=AF_INET;
local.sin_port=htons(6677);
local.sin_addr.s_addr=inet_addr(argv[1]);
if(bind(s,(struct sockaddr *)&local,sizeof(local))==-1)
{
perror("bind error");
exit(1);
}
bzero((char *)&servaddr,sizeof(local));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons((short)atoi(argv[2]));
servaddr.sin_addr.s_addr=inet_addr(argv[1]);
strcpy(buffer,"TIME");
if(sendto(s,buffer,sizeof(buffer),0,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
perror("error in sendto");
exit(0);
}
t=sizeof(servaddr);
printf("the current time is:");
if((n=recvfrom(s,buffer,1024,0,(struct sockaddr *)&servaddr,&t))>0)
{
buffer[n]='\0';
fputs(buffer,stdout);
}
else
{
if(n<0)
{
perror("error in read from");
exit(0);
}
else
printf("server closed connection\n");
exit(1);
}
memset(buffer,0,100);
close(s);
return 0;                              }
```

CONCURRENT SERVER PROGRAM – conserver.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<string.h>
#include<netinet/in.h>
#include<errno.h>
#include<time.h>
int main(int argc,char *argv[])
{
int s,t,cp;
struct sockaddr_in servaddr,cliaddr;
char buffer[1024];
time_t ct;
if(argc!=2)
{
printf("\n usage:client<server-adr><port>");
exit(0);
}
if((s=socket(AF_INET,SOCK_DGRAM,0))<0)
{
perror("error in socket creation");
exit(0);
}
bzero((char*)&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_port=htons((short)atoi(argv[1]));
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
if(bind(s,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
perror("bind");
exit(0);
}
t=sizeof(cliaddr);
memset(buffer,0,100);
while(1)
{
if(recvfrom(s,buffer,1024,0,(struct sockaddr*)&cliaddr,&t)<0)
{
perror("error in recvfrom");
exit(0);
}
ct=time(NULL);
sprintf(buffer,"%s",ctime(&ct));
if((cp=fork())==0)
{
while(1)
{
if(sendto(s,buffer,sizeof(buffer),0,(struct sockaddr*)&cliaddr,sizeof(cliaddr))<0)
{
perror("error in send to");
exit(0);
}
memset(buffer,0,100);
if(recvfrom(s,buffer,1024,0,(struct sockaddr*)&cliaddr,&t)<0)
{
perror("error in recvfrom");
exit(0);
}
sprintf(buffer,"%s",ctime(&ct));
}
}
else if(cp<0) {
perror("fork error");
exit(0);        }
}
close(s);
return 0;
}
```

OUTPUT

```
ifconfig
eth0      Link encap:Ethernet  HWaddr 54:be:f7:57:e8:c5
          inet addr:192.168.90.111  Bcast:192.168.90.255  Mask:255.255.255.0
          inet6 addr: fe80::56be:f7ff:fe57:e8c5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:36656 errors:0 dropped:31 overruns:0 frame:0
          TX packets:10582 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20471919 (20.4 MB)  TX bytes:1253533 (1.2 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:765 errors:0 dropped:0 overruns:0 frame:0
          TX packets:765 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:71838 (71.8 KB)  TX bytes:71838 (71.8 KB)

iit-b@inlabpc-11:~/Desktop/concurrent$ gcc conserver.c -o s
iit-b@inlabpc-11:~/Desktop/concurrent$ ./s 4011
```

CLIENT 1

```
gcc conclient.c -o c1
./c1 192.168.90.111 4011
the current time is:Mon Mar 19 11:09:27 2018
```

CLIENT 1 requests time – Server responds with current time

CLIENT 2

```
gcc conclient.c -o c2
./c2 192.168.90.111 4011
the current time is:Mon Mar 19 11:13:57 2018
```

CLIENT 2 requests time – Same Server responds with current time