Name :Umang

Reg. No. : 20214199

Group : CSE D

Computer Network Lab Assignment

1. The basic assignment is to write both an Echo Client and an Echo Server in C using UDPDatagram Sockets. The client and server implement the echo service while running ondifferent Linux machines and communicating with each other using UDP.

Server.c

```
//server.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <string.h>
#define ECHOMAX 255 // Define maximum echo buffer size
void DieWithError(const char *errorMessage); // Function prototype
int main(int argc, char *argv[]) {
int servSock;
struct sockaddr_in echoServAddr, echoClientAddr;
unsigned int cliAddrLen, recvMsgSize;
char echoBuffer[ECHOMAX];
if ((servSock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
DieWithError("socket() failed");
memset(&echoServAddr, 0, sizeof(echoServAddr));
echoServAddr.sin_family = AF_INET;
echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY);
echoServAddr.sin_port = htons(atoi(argv[1])); // Get port number from command line
if (bind(servSock, (struct sockaddr *)&echoServAddr, sizeof(echoServAddr)) < 0)
DieWithError("bind() failed");
for (;;) {
```

```
cliAddrLen = sizeof(echoClientAddr);
if ((recvMsgSize = recvfrom(servSock, echoBuffer, ECHOMAX, 0, (struct sockaddr
*)&echoClientAddr, &cliAddrLen)) < 0)
DieWithError("recvfrom() failed");
// Print the received message
echoBuffer[recvMsgSize] = '¥0'; // Null-terminate the string
printf("Received message: %s¥n", echoBuffer);
if (sendto(servSock, echoBuffer, recvMsgSize, 0, (struct sockaddr *)&echoClientAddr,
sizeof(echoClientAddr)) != recvMsgSize)
DieWithError("sendto() failed");
}
close(servSock);
return 0;
}
void DieWithError(const char *errorMessage){
perror(errorMessage);
exit(EXIT_FAILURE);
}
```

Client.c

```
//client.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <string.h>
void DieWithError(const char *errorMessage) {
perror(errorMessage);
exit(EXIT_FAILURE);
}
int main(int argc, char const *argv[]) {
if (argc != 3) {
fprintf(stderr, "Usage: %s <Server IP> <Server Port>¥n", argv[0]);
exit(EXIT_FAILURE);
}
int clientSock;
struct sockaddr_in echoServAddr;
unsigned int echoServPort;
const char *echoservIP = argv[1]; // Server IP from command line
const char *echoString = "Hello, server!";
int echoStringLen;
if ((clientSock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
DieWithError("socket() failed");
memset(&echoServAddr, 0, sizeof(echoServAddr));
echoServAddr.sin_family = AF_INET;
echoServAddr.sin_addr.s_addr = inet_addr(echoservIP);
echoServPort = atoi(argv[2]); // Server port from command line
echoServAddr.sin_port = htons(echoServPort);
echoStringLen = strlen(echoString);
if (sendto(clientSock, echoString, echoStringLen, 0, (struct sockaddr *)&echoServAddr,
sizeof(echoServAddr)) != echoStringLen)
DieWithError("sendto() sent a different number of bytes than expected");
close(clientSock);
return 0;
}
```

2. Write a server program which sends the time of day information to the client. Alsodevelop a client interface for interacting with the server. The client should first send arequest message to the server asking for the time of day information. The server in turnresponds to the client with its time of day information. The IP address and port number ofthe server is to be passed as command line arguments to the client code. Similar namingconventions, compilation and output commands are to be used as mentioned in theprevious question. Use proper display messages on both the client and server consoles. UseUDP sockets. (Naming Convention – TimeOfDayClient.c, TimeOfDayServer.c)

NOTE:The client and server codes can initially be tested locally using the loopback address 127.0.0.1, but eventually the codes are to be tested on two different machines.

```
//server.c
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 12346
#define BUFFER_SIZE 1024
int main() {
int sockfd;
struct sockaddr_in servaddr, cliaddr;
// Creating socket file descriptor
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
perror("socket creation failed");
exit(EXIT_FAILURE);
}
memset(&servaddr, 0, sizeof(servaddr));
memset(&cliaddr, 0, sizeof(cliaddr));
// Filling server information
servaddr.sin_family = AF_INET; // IPv4
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(PORT);
// Binding socket with server address
if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
perror("bind failed");
exit(EXIT_FAILURE);
}
int len, n;
char buffer[BUFFER_SIZE];
len = sizeof(cliaddr); //len is value/resuslt
while (1) {
n = recvfrom(sockfd, (char *)buffer, BUFFER_SIZE, MSG_WAITALL, (struct sockaddr
*)&cliaddr,
&len);
buffer[n] = '¥0';
printf("Client : %s¥n", buffer);
time_t rawtime;
struct tm *info;
time(&rawtime);
info = localtime(&rawtime);
strftime(buffer, BUFFER_SIZE, "%Y-%m-%d %H:%M:%S", info);
sendto(sockfd, (const char *)buffer, strlen(buffer), MSG_CONFIRM, (const struct
sockaddr
*)&cliaddr, len);
printf("Time sent to client.¥n");
}
return 0;
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#define PORT 8080
#define BUFFER_SIZE 1024
int main(int argc, char *argv[]) {
if (argc != 3) {
printf("Usage: %s <server_ip_address> <server_port>¥n", argv[0]);
exit(EXIT_FAILURE);
}
int sockfd;
struct sockaddr_in servaddr;
// Creating socket file descriptor
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
perror("socket creation failed");
exit(EXIT_FAILURE);
}
memset(&servaddr, 0, sizeof(servaddr));
// Filling server information
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(atoi(argv[2]));
if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0) {
perror("Invalid address/ Address not supported");
exit(EXIT_FAILURE);
}
char buffer[BUFFER_SIZE];
printf("Sending request to server for time of day information...¥n");
sendto(sockfd, "TimeRequest", strlen("TimeRequest"), MSG_CONFIRM, (const struct
sockaddr
*)&servaddr, sizeof(servaddr));
int n;
socklen_t len;
n = recvfrom(sockfd, (char *)buffer, BUFFER_SIZE, MSG_WAITALL, (struct sockaddr
*)&servaddr,
&len);
buffer[n] = '¥0';
printf("Time from server: %s¥n", buffer);
close(sockfd);
return 0;
}
```

****