**27/09/2021 CN SPOT 4 2019103573**

**NON – PERSISTENT CONNECTION**

**SERVER.C**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<string.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<netinet/in.h>

#define PORT 8369

#define BUF\_SIZE 2000

#define CLADDR\_LEN 100

void main() {

    struct sockaddr\_in addr, cl\_addr;

    int sockfd, len, ret, newsockfd;

    char buffer[BUF\_SIZE];

    pid\_t childpid;

    char clientAddr[CLADDR\_LEN];

    char send\_message[BUF\_SIZE] = "HTML/1.1 200 Done\nServer: localserver\nConnection: close

\nHello from server\n";

    char msg[BUF\_SIZE] = "GET/message:";

    char print\_message[BUF\_SIZE] = "http/1.1\nUser-agent: client\nConnection: close\n";

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if (sockfd < 0) {

        printf("Error creating socket!\n");

        exit(1);

    }

    printf("Socket created...\n");

    memset(&addr, 0, sizeof(addr));

    addr.sin\_family = AF\_INET;

    addr.sin\_addr.s\_addr = INADDR\_ANY;

    addr.sin\_port = PORT;

    ret = bind(sockfd, (struct sockaddr \*) &addr, sizeof(addr));

    if (ret < 0) {

        printf("Error binding!\n");

        exit(1);

    }

    printf("Binding done...\n");

    printf("Waiting for a connection...\n");

    listen(sockfd, 5);

    for (;;) {

        len = sizeof(cl\_addr);

        newsockfd = accept(sockfd, (struct sockaddr \*) &cl\_addr, &len);

        if (newsockfd < 0) {

            printf("Error accepting connection!\n");

            exit(1);

        }

        printf("Connection accepted...\n");

        inet\_ntop(AF\_INET, &(cl\_addr.sin\_addr), clientAddr, CLADDR\_LEN);

        if ((childpid = fork()) == 0) {

            close(sockfd);

for (;;) {

                memset(buffer, 0, BUF\_SIZE);

                ret = recvfrom(newsockfd, buffer, BUF\_SIZE, 0, (struct sockaddr \*) &cl\_addr, &len);

                if(ret < 0) {

                    printf("Error receiving data!\n");

                    exit(1);

                }

                printf("Received data from %s: %s\n", clientAddr, buffer);

                printf("%s",msg);

                printf("%s",buffer);

                printf("%s",print\_message);

                ret = sendto(newsockfd, send\_message, BUF\_SIZE, 0, (struct sockaddr \*) &cl\_addr, len);

                if (ret < 0) {

                    printf("Error sending data!\n");

                    exit(1);

                }

                printf("Sent data to %s: %s\n", clientAddr, buffer);

            }

        }

        close(newsockfd);

    }

}

**CLIENT.C**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<string.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<netdb.h>

#define PORT 8369

#define BUF\_SIZE 2000

int main(int argc, char\*\*argv) {

    struct sockaddr\_in addr, cl\_addr;

    int sockfd, ret;

    char buffer[BUF\_SIZE];

    struct hostent \* server;

    char \* serverAddr;

    if (argc < 2) {

        printf("usage: client < ip address >\n");

        exit(1);

    }

    serverAddr = argv[1];

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if (sockfd < 0) {

        printf("Error creating socket!\n");

        exit(1);

    }

    printf("Socket created...\n");

    memset(&addr, 0, sizeof(addr));

    addr.sin\_family = AF\_INET;

    addr.sin\_addr.s\_addr = inet\_addr(serverAddr);

    addr.sin\_port = PORT;

    ret = connect(sockfd, (struct sockaddr \*) &addr, sizeof(addr));

    if (ret < 0) {

        printf("Error connecting to the server!\n");

        exit(1);

    }

printf("Connected to the server...\n");

    memset(buffer, 0, BUF\_SIZE);

    printf("Enter your message(s): ");

    fgets(buffer, BUF\_SIZE, stdin);

    ret = sendto(sockfd, buffer, BUF\_SIZE, 0, (struct sockaddr \*) &addr, sizeof(addr));

    if (ret < 0) {

        printf("Error sending data!\n\t-%s", buffer);

    }

    ret = recvfrom(sockfd, buffer, BUF\_SIZE, 0, NULL, NULL);

    if (ret < 0) {

        printf("Error receiving data!\n");

    }

    else {

        printf("Received: ");

        fputs(buffer, stdout);

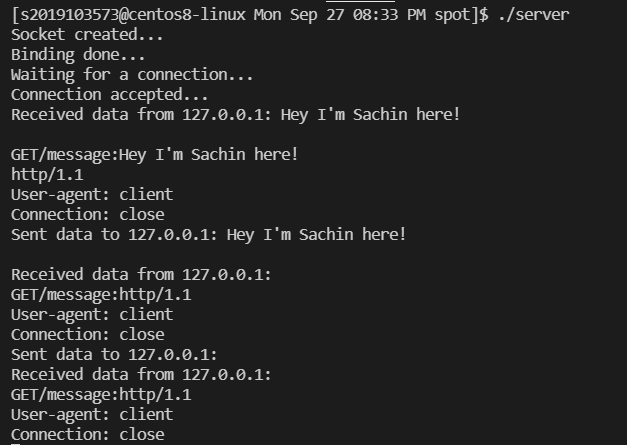
        printf("\n");

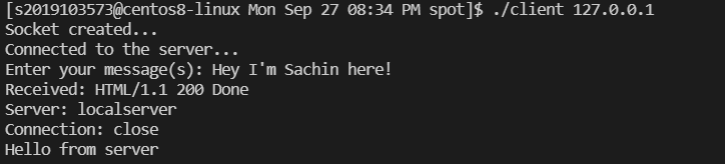
    }

    return 0;

}

**OUTPUT :-**

**SERVER**

**CLIENT**