27-09-2021 COMPUTER NETWORKS SACHIN RAGHUL T

LAB - 04 2019103573

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] = "HTTP/1.1 200 Done \n server:
  local server \n connection : close \n Hello from server \n";

  char msg[BUF_SIZE] = "GET /message;";
  char print_message[BUF_SIZE] = "http/1.1 \n User-agent: client
                                  \n connection : 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.sin_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, buffer, BUF_SIZE, 0, (struct sockaddr*)
                    &cl_addr, len);

        if (ret<0)
        {
          printf("Error sending data\n");
          exit(0);
        }
          printf("send 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, claddr;
    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) : ");

while (fgets (buffer, BUF_SIZE, stdin) != NULL)
{
  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:

Socket created...
Binding done...
Waiting for a connection...
connection accepted...
Received data from 127.0.0.1 : Heyy!

GET/message : Heyy!
http/1.1
User-agent : client
connection : close
Sent data to 127.0.0.1 : Heyy!

Received data from 127.0.0.1 : I am Sachin Raghul

GET / message : I am Sachin Raghul
http / 1.1
User-agent : client
connection : close
Sent data to 127.0.0.1 : I am Sachin Raghul

Received data from 127.0.0.1 : Bye Exiting

Client :

Socket created...
Connected to the server...
Enter your message(s) : Hayy!
Received : HTTP/1.1 200 Done
Server : localserver
connection : close
Hello from Server

I am Sachin Raghul
Received : HTTP/1.1 200 Done
Server : localserver
connection : close
Hello from Server.

Bye Exiting
Received : HTTP/1.1 200 Done
server : localserver
connection : close
Hello from Server.