Assignment 4

Ramana Srivats S

Anna University

1. Explain the connection procedure followed in client server communication

SETTING UP THE SERVER

1. **Socket Creation**

   int server_socket = socket(domain, type, protocol)

   The first step for setting up the server is to create a socket. A socket is nothing but an endpoint of a two-way communication between processes on the network. Here

   domain – This specifies the communication domain. For IPv4 we use AF_INET and for IPv6 we use AF_I NET 6

   type - communication type
   SOCK_STREAM: TCP
   SOCK_DGRAM: UDP

   protocol - Specifies a particular protocol to be used with the socket. For IP we use 0

2. **Bind**

   int bind(int server_socket, (struct_addr*) &server_adddress, socklen_t addrlen);

   After creating the socket, the next step is to bind the socket with the address and port number specified server_address and addrlen represents the size of server_address

3. **Listen**

   int listen(int server_socket, int backlog)

   This makes the server to listen for a client request to make a connection. backlog represents the maximum queue length the server_socket may grow.

4. **Accept**

   int client_socket = accept(int server_socket, struct sockaddr *addr, socklen_t *addrlen);

   It extracts the first connection request on the queue of pending connections for the listening socket, server_socket, creates a new connected socket, and

returns a new file descriptor referring to that socket. At this point, the connection is established between client and server, and they are ready to transfer data.

SETTING UP THE CLIENT

1. **Socket Creation**

   Similar to that of server socket creation

2. **Connect**

   int connect(int client_socket, struct sockaddr *addr, socklen_t addrlen);

   The connect system call connects the socket referred to by the file descriptor to the address specified by addr.

2. What is the use of bind() function in socket programming ?

   The bind() function is used to associate a socket with a specific network address (IP address and port number). This function is typically used on the server side to bind a socket to a specific port so that clients can connect to it.

   int bind(int server_socket, (struct_addr*) &server_adddress, socklen_t addrlen);

   The above system call binds the socket with the address and port number specified server_address and addrlen represents the size of server_address.

3. What is Datagram Socket?

   Datagram socket is a type of network socket that provides a way to send and receive data packets without a connection. They rely on User Datagram Protocol for packet transmission. Sending data packets this way means that it is less reliable (i.e.) there is no way to know whether the packet being transmitted is received by the receiver. But on the plus side we get higher speeds since we don't have to worry about acknowledgements.

4. Write a server/client model socket program to exchange hello message between them.

   SERVER

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
```

```
int main()
{

    //Creation of socket
    int server_socket = socket(AF_INET, SOCK_STREAM, 0);

    //Defining server address
    struct sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(9002);
    servAddr.sin_addr.s_addr = INADDR_ANY;

    // Binding the socket with address and port number
    bind(server_socket, (struct sockaddr*) &servAddr, sizeof(servAddr));

    // listening for connections
    listen(server_socket, 1);

    // Client socket
    int client_socket = accept(server_socket , NULL, NULL);

    //Sending a message to client
    char serMsg[1024] = “Hello Client”
    send(client_socket, serMsg, sizeof(serMsg), 0);
}
```

CLIENT

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>


int main()
{

    //Creation of socket
    int client_socket = socket(AF_INET, SOCK_STREAM, 0);

    //Defining server address
    struct sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;
```

```
    servAddr.sin_port = htons(9002);
    servAddr.sin_addr.s_addr = INADDR_ANY;

    int connectStatus
     = connect(sockD, (struct sockaddr*)&servAddr,
            sizeof(servAddr));

  if (connectStatus == -1) {
     printf("Error...\n");
  }

  else {
     char strData[1024];
     recv(sockD, strData, sizeof(strData), 0);
     printf("Message: %s\n", strData);
  }

}
```

5. Write a TCP server-client program to check if a given string is Palindrome

   Input: level

   Output: Palindrome

   Input: Assessment

   Output: Not a Palindrome

SERVER

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

int main()
{
    int left, right, flag;
    char b1[20];
    //Creation of socket
    int server_socket = socket(AF_INET, SOCK_STREAM, 0);
```

```
    //Defining server address
    struct sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(9002);
    servAddr.sin_addr.s_addr = INADDR_ANY;

    // Binding the socket with address and port number
    bind(server_socket, (struct sockaddr*) &servAddr, sizeof(servAddr));

    // listening for connections
    listen(server_socket, 1);

    // Client socket
    int client_socket = accept(server_socket , NULL, NULL);

    for (;;) {
     recv(client_socket, b1, sizeof(b1), 0);
     printf("\nThe string received is:%s\n", b1);
     if (strlen(b1) == 0)
        flag = 1;
     else {
        left = 0;
        right = strlen(b1) - 1;
        flag = 1;
        while (left < right && flag) {
           if (b1[left] != b1[right])
              flag = 0;
           else {
              left++;
              right--;
           }
        }
     }
     send(client_socket, &flag, sizeof(int), 0);
     break;
   }
}
```

CLIENT

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

int main()
{
    int flag;
    char buffer[20];
    //Creation of socket
    int client_socket = socket(AF_INET, SOCK_STREAM, 0);

    //Defining server address
    struct sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(9002);
    servAddr.sin_addr.s_addr = INADDR_ANY;

    int connectStatus
     = connect(sockD, (struct sockaddr*)&servAddr,
            sizeof(servAddr));

  if (connectStatus == -1) {
    printf("Error...\n");
  }

  else {
    for (;;) {
    printf("\nEnter a string to check palindrome: ");
    scanf("%s", buffer);

    printf("\nClient: %s", buffer);
    send(client_socket, buffer, sizeof(buffer), 0);
    recv(client_socket, &flag, sizeof(int), 0);

    if (flag == 1) {
      printf("\nServer: The string is a Palindrome.\n");
      break;
    }
    else {
```

```
            printf("\nServer: The string is not a palindrome.\n");
            break;
        }
       }
     }
   }
```

6. Write an example to demonstrate UDP server-client program.

CLIENT

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int main(int argc, char **argv){
 if (argc != 2) {
  printf("Usage: %s <port>\n", argv[0]);
  exit(0);
 }
 char *ip = "127.0.0.1";
 int port = atoi(argv[1]);

 int sockfd;
 struct sockaddr_in addr;
 char buffer[1024];
 socklen_t addr_size;

 sockfd = socket(AF_INET, SOCK_DGRAM, 0);
 memset(&addr, '\0', sizeof(addr));
```

```
  addr.sin_family = AF_INET;
  addr.sin_port = htons(port);
  addr.sin_addr.s_addr = inet_addr(ip);

  bzero(buffer, 1024);
  strcpy(buffer, "Hello, World!");
  sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&addr, sizeof(addr));
  printf("Data send: %s\n", buffer);

  bzero(buffer, 1024);
  addr_size = sizeof(addr);
  recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&addr, &addr_size);
  printf("Data recv: %s\n", buffer);

  return 0;
}
```

SERVER

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int main(int argc, char **argv){
  if (argc != 2){
    printf("Usage: %s <port>\n", argv[0]);
    exit(0);
  }
```

```
char *ip = "127.0.0.1";
int port = atoi(argv[1]);
int sockfd;
struct sockaddr_in server_addr, client_addr;
char buffer[1024];
socklen_t addr_size;
int n;
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockfd < 0){
  perror("socket error");
  exit(1);
}

memset(&server_addr, '\0', sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
server_addr.sin_addr.s_addr = inet_addr(ip);

n = bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
if (n < 0) {
  perror("bind error");
  exit(1);
}

bzero(buffer, 1024);
addr_size = sizeof(client_addr);
recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&client_addr, &addr_size);
printf("Data recv: %s\n", buffer);
```

```
  bzero(buffer, 1024);
  strcpy(buffer, "Welcome to the UDP Server.");
  sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&client_addr, sizeof(client_addr));
  printf("Data send: %s\n", buffer);
  return 0;
}
```