Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

MINOR ASSIGNMENT-12

Socket Programming: TCP & UDP (Networking Communication)

Practical Programming with C (CSE 3544)

Publish on: 24-12-2024 **Submission on:** 28-12-2024
Course Outcome: CO_6 **Program Outcome:** PO_4 **Learning Level:** L_5

Problem Statement:

Experiment with networking communication between client and server using socket API.

Assignment Objectives:

Students will be able learn how to establish communication between different programs (*i.e. processes*) over the network using TCP and UDP based socket programming.

Answer the followings:

1. Write C statements to create an IPV4 socket address structure and fill the structure with family **`AF_INET`**, **`port`**=34567 and **`IP`** address=127.0.0.0.1.

C Statements

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

int main() {
   struct sockaddr_in addr;

   memset(&addr, 0, sizeof(addr));
   addr.sin_family = AF_INET;
   addr.sin_port = htons(34567);
   if (inet_pton(AF_INET, "127.0.0.1", &addr.sin_addr) <= 0) {
       perror("inet_pton");
       exit(EXIT_FAILURE);
   }
   printf("Socket address created and initialized.\n");
   return 0;
}
```

2. Write C statements to declare **two** Internet socket address structure, namely **`servaddr`** and **`cliaddr`** respectively. Read the port and IP address for the structures from the keyboard and display the port and IP address onto the monitor.

C Statements

```
#include <stdio.h>
#include <arpa/inet.h>
#include <netinet/in.h>

int main() {
   struct sockaddr_in servaddr, cliaddr;
   char serv_ip[INET_ADDRSTRLEN], cli_ip[INET_ADDRSTRLEN];
   int serv_port, cli_port;

   printf("Enter server IP and port: ");
   scanf("%s %d", serv_ip, &serv_port);
   printf("Enter client IP and port: ");
   scanf("%s %d", cli_ip, &cli_port);
inet_pton(AF_INET, serv_ip, &servaddr.sin_addr);
   inet_pton(AF_INET, cli_ip, &cliaddr.sin_addr);
   servaddr.sin_family = cliaddr.sin_family = AF_INET;
   servaddr.sin_port = htons(serv_port);
   cliaddr.sin_port = htons(cli_port);

   printf("Server IP: %s\nServer Port: %d\nClient IP: %s\nClient Port:
%d\n",
       serv_ip, serv_port, cli_ip, cli_port);

   return 0;
}
```

3. Determine the output of the following code snippet and write the opening file/socket descriptors that are opened for the process using the command **`$ ls /proc/PID/fd`** in an another terminal.

```
int main(){
 int sockfd,count=0,i;
 printf("PID=%ld\n",(long)getpid());
 for(i=0;i<=9;i++){
   sockfd=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
   count=count+1;
   printf("%d ",sockfd);
 }
 printf("\nsocket descriptor count=%d\n",count);
 while(1);
 return 0;
}
```

Output

PID=12345 //Assuming that PID is 12345
3 4 5 6 7 8 9 10 11 12
socket descriptor count=10

4. Find out the output of the given code snippet and justify the reason of getting such output (**Hint: look into Host byte order and Network byte order**).

```
int main()
{
 struct sockaddr_in servaddr;
 servaddr.sin_family=AF_INET;
 servaddr.sin_port=htons(16);
 printf("Port given=%d\n",servaddr.sin_port);
 return 0;
}
```

Output

Output- Port given=4096
Reason- The sin_port field is stored in network byte order, but the printf function prints it as a decimal integer without converting it back to host byte order.

5. Find out the output of the given code snippet.

```
int main()
{
 struct sockaddr_in servaddr;
 servaddr.sin_family=AF_INET;
 servaddr.sin_port=16;
 printf("Port=%d\n",servaddr.sin_port);
 return 0;
}
```

Output

Port=16

This is because the value 16 is stored directly without any byte order conversion.

6. Fill out the missing parts of the following code snippet and Determine the output for the given port address as input: 16, 67, 879 respectively.

```
int main()
{
   __________ port;
   printf("Enter a port address:");
   scanf("%____",&port);
   struct sockaddr_in servaddr;
   servaddr.sin_family=AF_INET;
   servaddr.sin_port=htons(port);
   printf("Port=%d\n",htons(servaddr.sin_port));
   return 0;
}
```

Output

```
#include <stdio.h>
#include <arpa/inet.h>
#include <netinet/in.h>
int main(){
   int port;
   printf("Enter a port address:");
   scanf("%d", &port);
   struct sockaddr_in servaddr;
   servaddr.sin_family =
AF_INET;
   servaddr.sin_port = htons(port);
   printf("Port=%d\n",
   ntohs(servaddr.sin_port));
   return 0;
}
```

OUTPUT
Enter a port address: 16
Port=16

Enter a port address: 67
Port=67

Enter a port address:
879 Port=879

7. Develop a TCP based client-server programs to establish a communication between the client program and the server program as:
   - Server: sending data to the client
   - Client: Reading data that has been sent from the server

TCP Server Code here

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
   int server_fd, new_socket;
   struct sockaddr_in address;
   int opt = 1;
   int addrlen = sizeof(address);
   char *message = "Hello from server!";
   char buffer[BUFFER_SIZE] = {0};

   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
      perror("socket failed");
      exit(EXIT_FAILURE);
   }
   if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt))) {
      perror("setsockopt");
      close(server_fd);
      exit(EXIT_FAILURE);
   }
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons(PORT);

   if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
      perror("bind failed");
      close(server_fd);
      exit(EXIT_FAILURE);
   }
   if (listen(server_fd, 3) < 0) {
      perror("listen");
      close(server_fd);
      exit(EXIT_FAILURE);
   }

   if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
      perror("accept");
      close(server_fd);
      exit(EXIT_FAILURE);
   }

   send(new_socket, message, strlen(message), 0);
   printf("Message sent: %s\n", message);

   close(new_socket);
   close(server_fd);
   return 0;
}
```

TCP Client Code here **Specify: input & output**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation error");
        exit(EXIT_FAILURE);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        perror("Invalid address/ Address not supported");
        exit(EXIT_FAILURE);
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Connection Failed");
        exit(EXIT_FAILURE);
    }

    read(sock, buffer, BUFFER_SIZE);
    printf("Message received: %s\n", buffer);

    close(sock);
    return 0;
}
```

8. Develop a UDP based client-server programs to establish a communication between the client program and the server program as:
   - Server: sending data to the client
   - Client: Reading data that has been sent from the server

UDP Server Code here

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sockfd;
    struct sockaddr_in servaddr, cliaddr;
    char buffer[BUFFER_SIZE];
    char *message = "Hello from server!";
    socklen_t len;
    int n;

    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));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    len = sizeof(cliaddr);

    n = recvfrom(sockfd, (char *)buffer, BUFFER_SIZE, MSG_WAITALL, (struct sockaddr *)&cliaddr, &len);
    buffer[n] = '\0';
    printf("Client: %s\n", buffer);

    sendto(sockfd, (const char *)message, strlen(message), MSG_CONFIRM, (const struct sockaddr *)&cliaddr, len);
    printf("Message sent.\n");

    close(sockfd);
    return 0;
}
```

UDP Client Code here **Specify: input & output**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
   int sockfd;
   struct sockaddr_in servaddr;
   char buffer[BUFFER_SIZE];
   char *message = "Hello from client!";
   socklen_t len;

   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
      perror("socket creation failed");
      exit(EXIT_FAILURE);
   }

   memset(&servaddr, 0, sizeof(servaddr));

   servaddr.sin_family = AF_INET;
   servaddr.sin_port = htons(PORT);
   servaddr.sin_addr.s_addr = INADDR_ANY;

   len = sizeof(servaddr);

   sendto(sockfd, (const char *)message, strlen(message), MSG_CONFIRM, (const struct sockaddr
*)&servaddr, len);
   printf("Message sent: %s\n", message);

   int n = recvfrom(sockfd, (char *)buffer, BUFFER_SIZE, MSG_WAITALL, (struct sockaddr *)&servaddr,
&len);
   buffer[n] = '\0';
   printf("Server: %s\n", buffer);

   close(sockfd);
   return 0;
}
```